INTERFEROMETRIC MODIFICATION OF THE LOCKHEED MARTIN PSTAR SYSTEM TO FACILITATE THREE DIMENSIONAL AIRSPACE SURVEILLANCE

By

Scott E. Otterbacher

Recommended:

Advisory Committee Chair

Chair, Department of Electrical and Computer Engineering

Approved:

Dean, College of Engineering and Mines

Dean of the Graduate School

Apr 5, 2011

Date

INTERFEROMETRIC MODIFICATION OF THE LOCKHEED MARTIN PSTAR SYSTEM TO FACILITATE THREE DIMENSIONAL AIRSPACE SURVEILLANCE

A

THESIS

Presented to the Faculty

of the University of Alaska Fairbanks

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

By

Scott E. Otterbacher, B.S.

Fairbanks, Alaska

May 2011

Abstract

The Lockheed Martin PSTAR is a monostatic radar system that provides range, azimuth, and radial velocity information of detected targets. While this system is useful for airspace surveillance in remote locations due to its portability and durability, it lacks the ability to record target information and the ability to estimate target elevation angle, resulting in a vertical arc of possible target locations. Due to a desire to use the PSTAR for applications that require logging three-dimensional target information, a spatial interferometric modification has been implemented.

The PSTAR estimates range from pulse propagation delay and azimuth angle from the orientation of the antenna on a rotating pedestal. Two PSTAR antennas were removed from their housings and mounted, vertically separated, in a custom enclosure allowing for the estimation of elevation angle through spatial interferometry. The reflected signal is received by both antennas, mixed to baseband, and then the two pairs of I/Q channels are simultaneously sampled at 1 MS/s. Target elevation angle is estimated by determining the phase difference of the target's reflection received by the two vertically spaced antennas. Range, azimuth, and radial velocity are also estimated. All data collection was implemented in LabVIEW and data post-processing was implemented in MATLAB.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Unmanned Aerial Vehicles

Unmanned Aerial Vehicle (UAV) technology has progressed much in recent years and there are now a wide variety of UAVs produced commercially. These range from small craft, such as the hand launched AeroVironment Raven [1] (or smaller) up to the Northrop Grumman Global Hawk, which has a wingspan of 40 m [2]. While the number of UAV systems produced continues to increase, the operation of these systems within the United States remains dominated by the military [3]. This is not due to the lack of commercial applications for UAVs, but rather the lack of appropriate regulations regarding commercial UAV usage as well as the difficulty in obtaining permits to fly UAVs commercially.

UAVs differ from standard aircraft in that they do not have a human pilot aboard and differ from Remote Controlled (RC) aircraft in that there is not a human remotely operating the aircraft through direct control of the motor and maneuvering flaps. A UAV is flown by providing a computer instructions of the mission to be flown. These instructions can take a variety of forms, but a simple example is of a UAV flying a predetermined route. The UAV determines its location, often via an on-board GPS system, and the computer calculates the direction it needs to move to follow the flight plan. The computer then controls the motor and maneuvering flaps in such a way that the UAV follows the intended course at the intended velocity. Human interaction with the system is required whenever a deviation from the flight plan is required that the UAV computer is unaware of.

There are currently many roles being performed by manned aircraft that could be accomplished autonomously with UAVs. Some these are government-contracted jobs such as environmental monitoring, search and rescue, emergency response and monitoring, and border patrol. However, there also exist many purely commercial applications such as aerial photography, industrial surveillance, and agricultural usage such as monitoring and spraying of crops [4]. Transitioning from manned aircraft to UAVs fulfilling these duties has multiple benefits. One of the main benefits, especially to commercial interests, is the possible reduction in costs. This is due to factors including a decrease in aircraft size, and in turn fuel usage, as well as enabling longer duration flights. Additionally, risk to

pilots due to aircraft failure, or human error, is removed. An example of the successful integration of UAVs into the commercial sector can be found in the widespread use of the Yamaha RMAX autonomous helicopter in Japan for agriculture spraying. This system as been in use for over 10 years and has been readily adopted by commercial farms due to cost savings over traditional manned aircraft based spraying and the existence of Japanese legislation allowing for commercial UAV flights [5].

Not only are there opportunities for the replacement of current manned aircraft operations with UAVs, but there are also applications for UAVs that are not realizable using manned aircraft. An example of this are cases where the aircraft would be put into a hazardous situation, which may not be appropriate for a manned plane. This includes tasks such as wildfire monitoring where visibility is low and fire induced winds are unpredictable [6]. Putting a pilot in this situation may be considered too high of a risk, but a UAV may be chosen because the risks only involve a loss of equipment and not of human life. There are also tasks such as in situ volcanic plume measurements where the aircraft is expected to be destroyed during the course of the mission and would obviously not be an acceptable task for by a manned aircraft [7]. Enabling commercial access to UAV flights will undoubtedly result in new applications being found that are currently not being addressed by any existing systems.

1.2 FAA Requirements

All flights taking place within U.S. airspace are regulated under Federal Aviation Administration (FAA) jurisdiction as granted by law [8]. Existing FAA collision avoidance protocols are based on visual contact between aircraft [9]. This presents a roadblock for commercial UAV usage due to the limited visual awareness of surrounding airspace without the use of a ground spotter or a manned chase plane. Additionally, many UAVs are considerably smaller than manned aircraft, which adds to the difficulty faced by a pilot to see and avoid the UAV. The two current sense and avoid options for UAV flight both have major drawbacks. The use of a ground spotter severely limits the range a UAV can operate over. One of the benefits of some UAVs over traditional manned aircraft is the increased flight duration that allows for increased spatial coverage; having a ground spotter in visual contact at all times eliminates this benefit, and is not possible in poor visibility conditions such as

wild fire monitoring. The use of a chase plane does allow for long distance operations, but flight time is still limited by the manned chase plane. The use of a chase plane also adds a large financial cost to the operation. Following existing FAA collision avoidance protocols that mandate human visual situational awareness places an operational and financial burden on the operators of UAVs that would otherwise not exist if an autonomous sense and avoid system were available and approved for use by the FAA.

There are two ways to get approval from the FAA to fly a UAV. The first, if the aircraft is civil, is to have the UAV platform pass the FAA Airworthiness Certification. This is a certification that all civil aircraft, manned or unmanned, must obtain to be legally operated in U.S. airspace. The main goal of this certification is to ensure a high level of confidence that the aircraft will not fail in a way that could cause harm to those aboard or those on the ground. While risk to passengers is not an issue with UAVs, at least currently as there are no unmanned passenger planes, there is still the risk to people and property on the ground if the UAV were to crash. The process for obtaining Airworthiness Certification for an aircraft involves a systems analysis of the aircraft as well as the consideration flight test data and failure rates [10]. This step is currently achievable for UAVs, however it may be inappropriate for small, low speed, low altitude UAVs that pose minimal risk to person and property in the event of a crash. The second approach suitable for public aircraft involves obtaining a Certificate of Authorization (COA) from the FAA for the specific use of a UAV where the public entity has already deemed the aircraft airworthy. The main item that must be addressed to obtain a COA is the demonstration that the UAV operation will pose minimal risk to person and property, both on the ground and in the air [11].

The definition of collision risks necessary to apply for a COA can be a difficult task. This is because the collision risk is currently unquantifiable in situations where there is an absence of air traffic data. This is an important risk to consider in Alaska due to the relatively large number of private aircraft as well as the fact that many do not carry transponders. The FAA wishes to gather flight statistics in areas where UAVs may potentially be operated with the goal of quantifying collision risks. The ability to gather flight statistics over long periods of time would also be useful to commercial entities interested in flying their certified aircraft. The goal of this thesis is to provide an instrument that can gather and store airspace flight statistics over long time durations to conduct such safety analyses.

To monitor air traffic, the Lockheed Martin Portable Search and Target Acquisition Radar (PSTAR) system was chosen due to its portability and the ability to obtain the systems for free from NASA Ames and the FAA. The PSTAR system provides range and azimuth information for detected targets. Since the PSTAR does not obtain altitude information it cannot provide the FAA with detailed enough statistics about flight patterns in a given area to allow for the quantification of collision risks. To accomplish the goal a PSTAR was modified to determine elevation angle in addition to the range and azimuth information the stock PSTAR provides. The three dimensional modification to the PSTAR provides a low cost solution for mobile airspace monitoring currently only feasible using expensive systems such as the Raytheon Sentinel radar.

An issue arising from the operation of a radar system is the need for FCC clearance to transmit. During development, clearance was obtained to operate the PSTAR at Poker Flat Research Range (PFRR) on multiple frequency bands. PFRR is located in a remote area and has many surrounding hills that block transmissions, so the chance of the PSTAR interfering with other systems is minimal. In more heavily populated areas it may be difficult to obtain clearance to transmit, especially as the PSTAR transmit on a band reserved for military use.

1.3 PFRR Requirements

In addition to providing the FAA with an instrument to document airspace flight statistics there are other roles that the system described within this thesis can fill. One is to use the system for sense and avoid. While current FAA policy does not routinely allow a radar system to replace a chase plane or a ground observer the radar system can assist UAV operators in situational awareness. The radar system is able to detect targets at distances unattainable by human eyes and allow for an earlier diversion of the UAV from its original flight plan to ensure it is kept at a safe distance from other aircraft. Additionally, a system such as the one described here provides an instrumental basis for the FAA to develop sense and avoid policies for UAVs that do not require human visual observation. The first non-FAA use of this system will likely be PFRR located near Chatinika, AK. PFRR currently operates an InSitu ScanEagle UAV (Figure 1.1) that has been used for applications ranging from wild fire monitoring to aerial imaging in the Bering Sea.

Figure 1.1. PFRR InSitu ScanEagle being launched. Photo Credit: PFRR.

A specific example of usage is a population count of a potentially endangered species. The count is facilitated by using the quieter UAV (Figure 1.2) over the traditional noisy helicopter which can cause the animals to flee.

PFRR has another use for the instrument described in this thesis: airspace surveillance during rocket launches. PFRR routinely launches sounding rockets for scientific studies and there is the safety requirement that the surrounding airspace be clear of aircraft during a launch. PFRR recently began using the two dimensional PSTAR, with modifications to allow data logging and enhanced display, as the airspace surveillance instrument during rocket launches. The three-dimensional modification described here are a useful upgrade to that system.

1.4 Thesis Overview

The following chapter will describe the operation of the unmodified PSTAR system and characterization testing that was performed on it. Chapter 3 provides an overview of the estimation of target parameters. Chapter 4 discusses the technique for capturing voltage-level receiver data from the PSTAR and the method for capturing azimuth information. Chapter 5 presents the algorithms that process the raw data to determine range, azimuth,

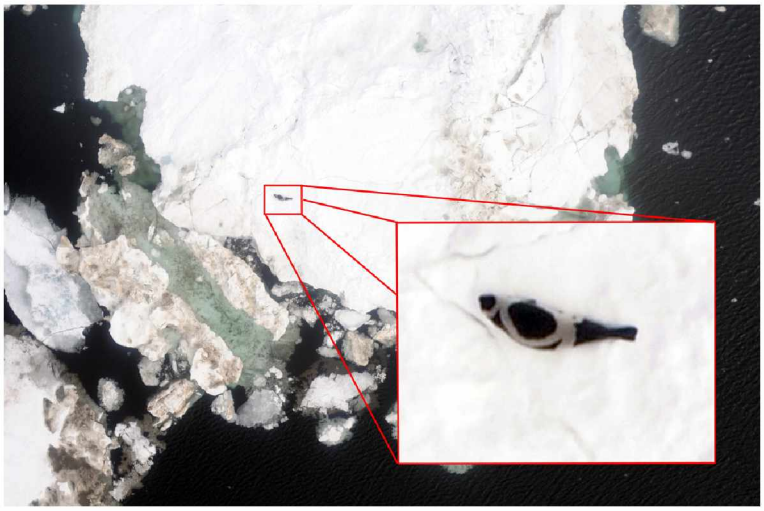

Figure 1.2. Aerial image of a ringed seal captured from the PFRR ScanEagle UAV over the Arctic Ocean. Photo Credit: PFRR

elevation, and radial velocity as well as a discussion of the errors inherent in the algorithms. Chapter 6 describes the testing that was performed to characterize the operation of the new system and a comparison of those test results to theoretical predictions. Chapter 7 investigates future improvements to the system. Algorithm software code can be found in the Appendix.

Chapter 2
Description of PSTAR

2.1 PSTAR Hardware

This section discusses details of the PSTAR as they are detailed in the U.S. Army Operator's Manual [12]. The Lockheed Martin PSTAR system is a two dimensional monostatic portable RADAR that operates in the L-band. The PSTAR was originally developed to provide airspace situational awareness to forward deployed military personnel, and is still in use by the United State's military as well as other countries [13]. The PSTAR was well designed for its intended role; it is rugged, easily portable, quick to set up and tear down, has a easily read display, and can be connected to various weapons systems such as surface to air missile batteries. However, the unmodified PSTAR is unable to meet all the requirements for the work described in this thesis. This chapter will describe the unmodified PSTAR system and the testing required to validate it.

The complete PSTAR system is pictured in Figure 2.1, however, not all of the components shown are used in the system described in this thesis.

Figure 2.1. PSTAR System [14].

The two omni-directional antennas on either side of the PSTAR are used to remove ambient RF interference from the signal received in the main antenna. This feature is primarily for jamming mitigation and will not be used in the three-dimensional system. The large box on the right side of the image is the Identify Friend or Foe (IFF) interrogator.

Because of the non-military nature of the three-dimensional system described here, this feature is also not used. The Command Interface Unit (CIU) is located on top of the cable spool. This is used to control the PSTAR system as well as display status and target information. A diagram showing the interconnections of the PSTAR components can be seen in Figure 2.2.

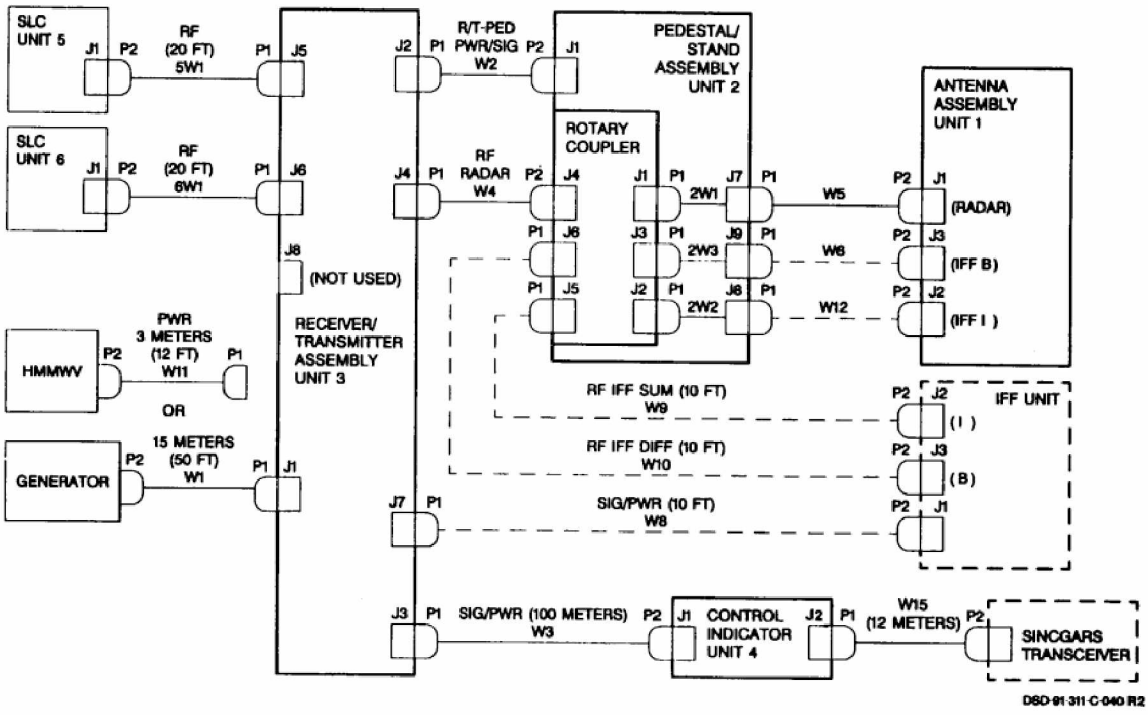

Figure 2.2. Connection diagram for the PSTAR [12].

Due to the limitations of the display and lack of data logging, the CIU is only used for control of the PSTAR system in the three-dimensional implementation. All display and data logging will be handled elsewhere by the modified system.

The PSTAR transceiver assembly is made up of five subsystem modules: receiver, stable master oscillator (STAMO), power amplifier, power supply, and processor. Each of these subsystems is rack-mounted inside the transceiver assembly with all interconnections between subsystems made on the back of the unit as shown in Figure 2.3. The division of the transceiver assembly into distinct easily replaceable subsystems is a useful design feature of the PSTAR, allowing for faulty subsystems to be readily swapped out

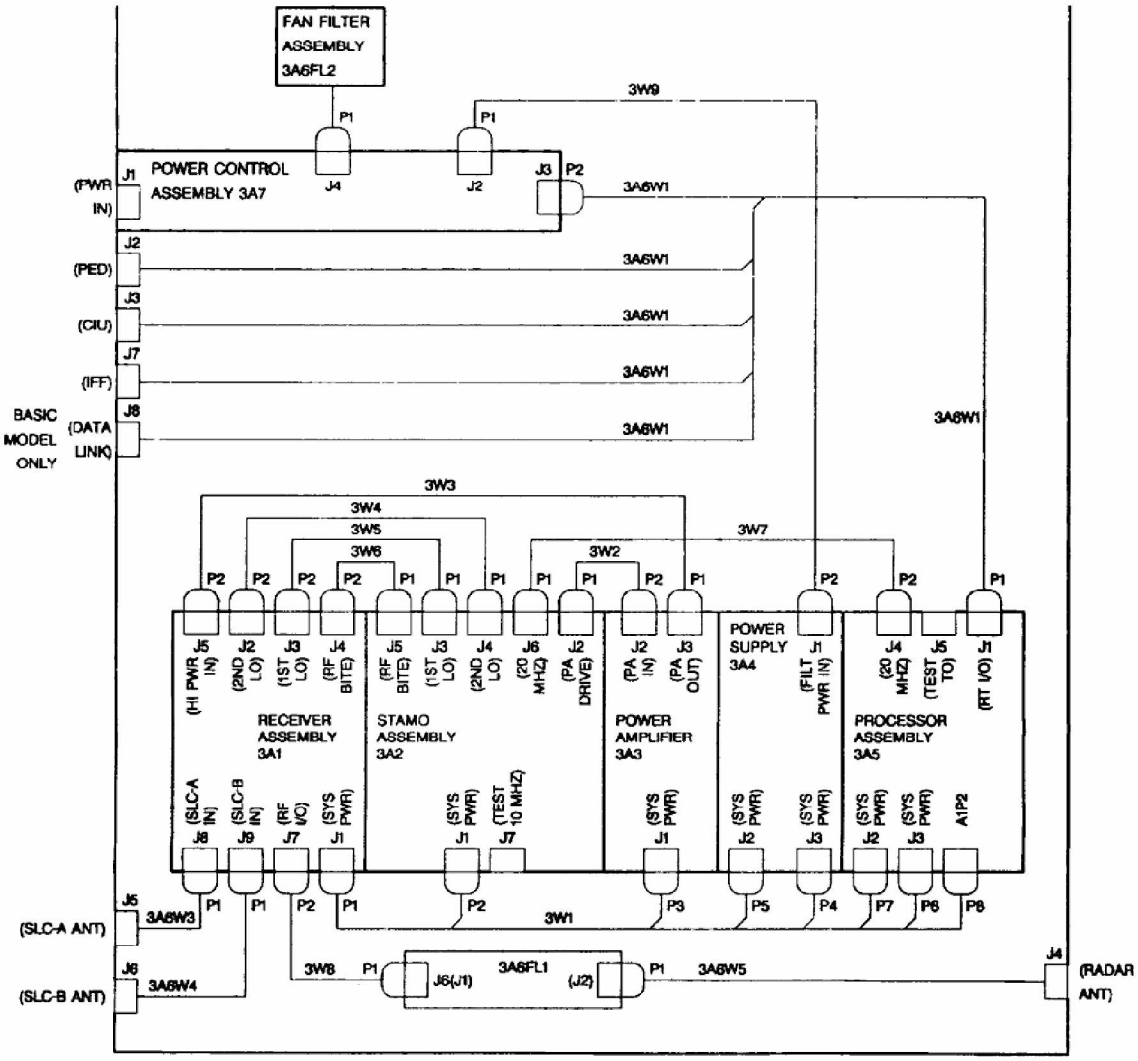

Figure 2.3. Block diagram of PSTAR subsystems [15].

for functional ones. The PSTAR transmits 1 kW peak power in a frequency range of 1.22-1.4 GHz, with 19 channels separated by 10 MHz each. The transmit frequency can be set to a single channel or can be set in agile frequency mode. This mode randomly switches at a rate of 15 Hz through a user-definable set of frequencies and is intended as an anti-jamming counter measure. The transmit pulse envelope is 8 μs in duration and contains a sinusoid of constant amplitude and frequency.

The PSTAR antenna is mounted on a rotating pedestal and is used for both transmit

and receive. The antenna scans a full 360° in about 6 seconds. The antenna produces a horizontally polarized fan beam with a vertical 3 dB beam width of 28° and a horizontal 3 dB beam width of 10.8°. Due to the azimuthal sampling rate, an azimuth resolution of 8° is achieved. The boresight gain of the antenna is 19.5 dBi. The antenna can be installed in either look up or look down mode. In look up mode the vertical beam pattern extends from the horizon to 28° above the horizon, while in look down mode the vertical beam pattern starts 5° below the horizon and extends to 23° above it.

The PSTAR uses a dual heterodyne receiver that converts the L-band carrier down to baseband before it is digitized. The receiver has a bandwidth of 1 MHz and a noise figure of 2 dB. The receiver has three channels, one for the main antenna and the remaining two for the omni-directional antennas. The two channels for the omni-directional antennas are Side-Lobe Cancellers (SLC) designed to mitigate interference. The three channels are shown as identical in the functional block diagram (Figure A.21), except for the addition of a bandpass filter on the main channel. However, it was determined through testing that the gain differs between the Main and SLC channels. The receiver is blanked during the transmit pulse and automatically attenuates signals at times immediately following. This is due to reflected signal strengths from close targets being much larger than from far targets. This attenuation is done in steps with a maximum of 26 dB immediately following the transmit pulse and no attenuation at the time corresponding to a range of 6 km. The relative timing of the attenuation steps and the transmit pulse is shown in Figure 2.4.

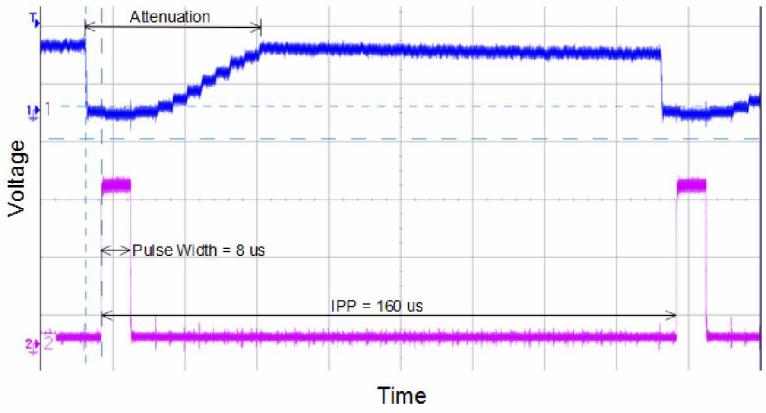

Figure 2.4. Top: Receiver attenuation steps. Bottom: Transmit pulse envelope.

Digitization of the base band signal is accomplished in the PSTAR processor module at 1 MS/s. Processing yields 32 range gates with 50% overlap of adjacent gates. The range resolution, determined by the length of the transmit pulse, is 1500 m and the range accuracy of the system is 200 m. The maximum range of the PSTAR is 20 km. Target Doppler velocity information is determined by the PSTAR system with a range of 20 – 550 m/s. All targets with a Doppler velocity of less than 20 m/s are not displayed to reduce the amount of stationary ground clutter. However, helicopters are displayed down to 0 m/s due to the Doppler spectrum associated with the blade flash. The Doppler resolution of the PSTAR system is unknown due to a lack of knowledge of the number of points the PSTAR uses when performing the discrete Fourier transform. A summary of the PSTAR system parameters is shown in Table 2.1.

Table 2.1. PSTAR parameters as listed in Lockheed Martin Operator's Manual [12].

Frequency	1.22-1.4 GHz
Transmit Power	1 kW
Pulse Width	8 μs
PRF	5.55/6.25 kHz
Dwell Time	384 Pulses
Antenna Gain	19.5 dBi
Polarization	Horizontal
Azimuth Beam Width	10.8°
Elevation Beam Width	28°
Scan Rate	10 rpm
Receiver Noise Figure	2 dB
Receiver Bandwidth	1 MHz
Max Range	20 km
Range Resolution	1500 m
Range Accuracy	200 m
Doppler Velocity	20-550 m/s
Azimuth Resolution	8°

2.2 PSTAR Testing

Before work towards modifying the PSTAR could begin, the available systems had to be tested to determine if they performed up to original specifications. This proved to be an important task due to the history of the available PSTAR units. Obtaining new mint-condition PSTARs for this project was not an option as the PSTAR is an expensive piece of military hardware to buy new from Lockheed Martin. In fact, the PSTAR was chosen based on the availability of military surplus units for low cost to the project. Being military surplus, the systems obtained had seen plenty of use and not all were in top condition, including some that were not operational. Testing of the PSTAR systems facilitated the determination of which individual subsystems were causing faults. Through the combination of subsystems from multiple PSTAR systems, working units were assembled and verified.

The most basic test was simply to assemble the systems and turn them on. The PSTAR has a Built In Test (BIT) that is run automatically at start-up and periodically during operation. The BIT operates by injecting an artificial target signal into the receiver and then comparing the resulting data from the processor module with existing data that is expected from a functional unit. If the BIT determines that the PSTAR is not functioning properly a fault is displayed on the CIU. Faults are also displayed on the CIU when connections between subsystems are not present, or when other types of operational errors are detected.

At the start of this project three PSTAR systems were acquired by the University of Alaska Fairbanks (UAF). Initial inspection, assembly, and operation of these units resulted in a variety of hardware issues being discovered. Inspection of the equipment showed that the majority of the cables were in poor condition with cracked insulation, rusty connectors, and loose connectors. All RF cables were replaced to improve signal quality. Working data and power cables were saved, but a number had to be replaced due to discontinuities in some of the connections. This included two of the three transceiver-pedestal cables.

Powering up of the units was initially attempted using one of the diesel generators that was supplied with the PSTAR. After a change of oil and fuel and a mechanical inspection the generator's internal combustion engine was operational. However, the electric generator connected to the engine failed to supply any power to the PSTAR. To power the PSTAR a 27 V 40 A power supply (IOTA DLS-27-40) was purchased and the PSTAR power cable was modified to connect to it. All three PSTAR units were powered up and inspected for

problems. One of the CIUs gave a CMOS checksum error and would not boot up properly. One transmitter produced a Voltage Standing Wave Ratio (VSWR) fault indicating an impedance mismatch in the transmit RF pathway. This caused the transmitter to be shut off automatically shortly after being powered on. One STAMO produced a fault at a couple different frequencies indicating that the frequency being produced was not correct. Two of the processor modules produced Signal to Noise Ratio (SNR) faults indicating that either the signal strength of the BIT signal was lower than expected or the system noise was higher. The systems with these processor modules also indicated that jamming was present at inconsistent azimuths during operation. This was most likely due to the poor SNR. After these initial tests were completed one satisfactorily functional PSTAR was assembled from the working subsystems of the original three.

Because the PSTAR systems are to be used differently than originally intended it was important to make detailed characterizations of the transmitters and receivers beyond what is accomplished with the BIT. The transmitters were first examined to determine four main performance parameters: peak power out, pulse width, Pulse Repetition Frequency (PRF), and carrier frequency. This was accomplished using the test setup shown in Figure 2.5.

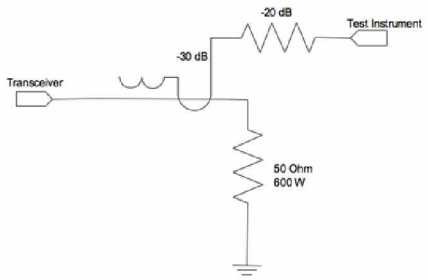

Figure 2.5. Transmitter test setup.

The power out was measured by connecting a power meter to the 20 dB coupler, the pulse width and PRF were measured by connecting an oscilloscope to the coupler, and the carrier frequency was checked with a spectrum analyzer. It was determined that the transmitters for all PSTARs were operating correctly. It was discovered through this testing that when operating in alternating PRF mode (the default) the PSTAR transmits 384 pulses

at each PRF. This is an indication that 384 pulses are used per dwell in computing target parameters in the PSTAR, however it is not definitively known if this is the case.

The proper functioning of the receivers is a key part in the operation of the three dimensional PSTAR described in this thesis. In normal operation the PSTAR receiver is supplied with an input of the signal received on the antenna. If a reflection off a target of the transmitted pulse is received the signal will be an attenuated copy of the transmit pulse with the carrier frequency slightly Doppler shifted due to the target's velocity. The signal is immediately amplified by a Low Noise Amplifier (LNA). After the LNA the signals goes through an adjustable attenuation stage. This may actually be adjustable gain, but this is unknown and is irrelevant. The amount of attenuation is decreased as time from the transmit pulse increases. The signal is then mixed with a sinusoid of frequency 300 MHz higher than the transmit frequency. The resulting signal is mixed in an in-phase and quadrature mixer with a 300 MHz sinusoid. This results in two 90° phase shifted baseband signals, I and Q channels, with a frequency equal to the Doppler shift of the original received signal.

It was discovered that the three PSTAR receivers at UAF were not functioning properly. While it was impossible to determine if a particular receiver was functioning as intended due to a lack of test requirements from Lockheed Martin, the three receivers have been compared with each other as well as with an estimate of what the output should be. Each receiver has three nominally identical channels (Main, SLC1, SLC2); however these do not all behave identically. There are four main issues that have arisen in testing: DC offset, I/Q imbalance (amplitude and phase), low amplitude, and erroneous attenuation steps.

There are two types of DC offsets involved. The first has to do with what voltage level the signal drops to during the transmit pulse. Ideally, this would be 0 V, however this is not so on all of the receivers. This is shown in Figure 2.6 by the 'line' at about -0.04 V.

This type of DC offset, while unwanted, does not pose a serious problem because the ADC will not be taking data during the transmit pulse. The other type of DC offset, also visible in Figure 2.6, is the center of the sinusoid not being at 0 V. This particular receiver shows a DC offset of about +0.01 V.

Each of the receiver channels has an I and Q channel. It is important that these channels be identical, but shifted in phase 90°. An amplitude imbalance in I/Q channels can be seen in Figure 2.7.

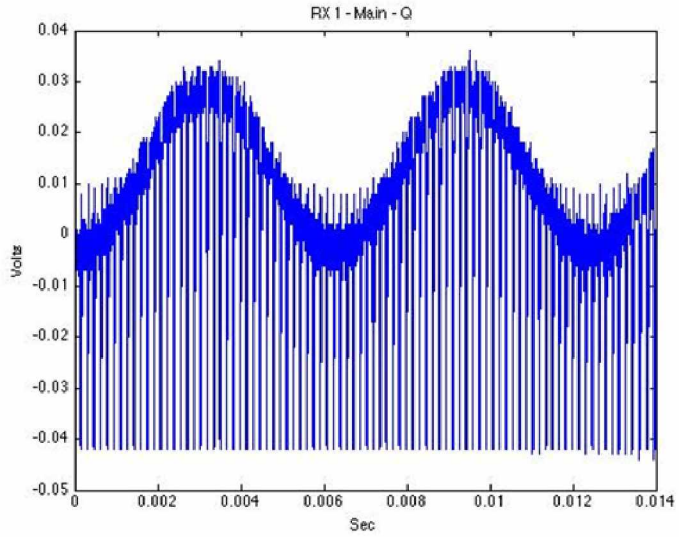

Figure 2.6. Receiver DC offset.

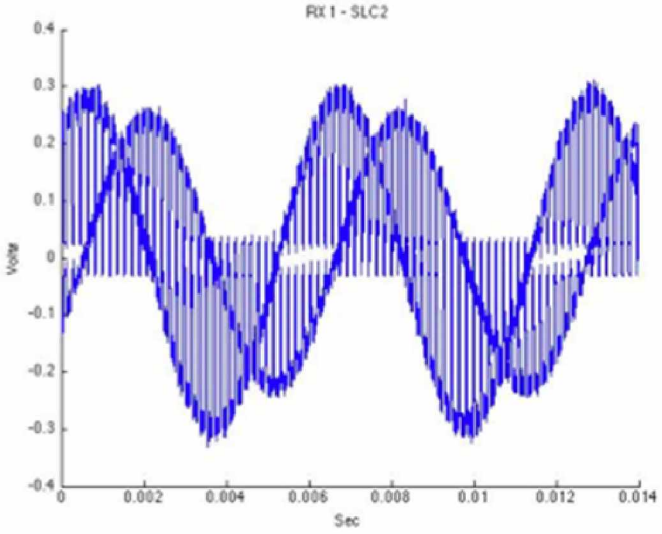

Figure 2.7. I/Q amplitude imbalance.

Previously, in Section 2.1, the attenuation scheme was discussed (Figure 2.4). Examples of erroneous attenuation can be seen in Figure 2.8.

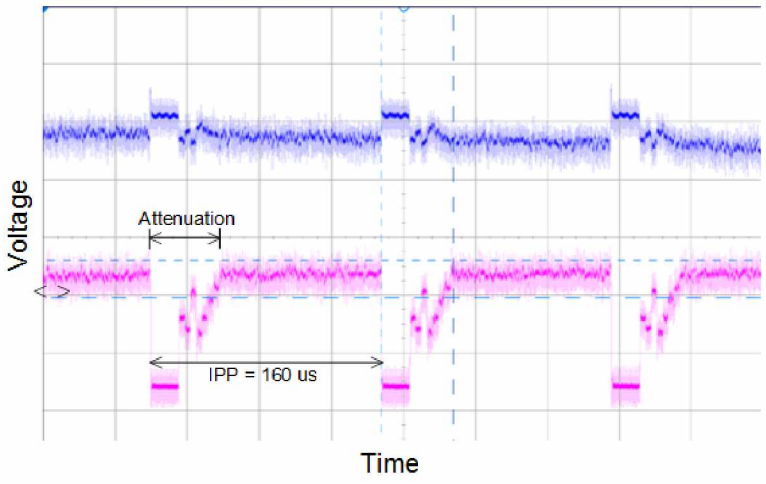

Figure 2.8. Erroneous attenuation steps.

Erroneous attenuation that is consistent between I and Q channels is undesired, but usable if need be. Attenuation that is not consistent between channels will affect the detection algorithm, and should not be used if at all possible, as was the case with one of the receivers.

The optimum solution to these problems is to get a pristine receiver, but it is possible to work around some of the problems by making adjustments to collected data before processing. Fixing the hardware could possibly have been done through a combination of reverse engineering and trial and error. However, it was decided to triage the receivers and use only those that are functioning properly. Some offsets and imbalances are still present in the good receivers and the method to mitigate this is discussed in Chapter 7.

With only one functional PSTAR from the original three provided to UAF more working units were desired, and three more were obtained from NASA Ames. The acquisition of these units involved testing five PSTARs at NASA Ames and assembling three usable units from the subsystems of the five. Testing was also performed on four PSTARs owned by the FAA in Atlantic City, NJ. Three of the four FAA units were shown to be functioning properly, and the forth had problems only in the SLC channels of the receiver making it usable for normal two-dimensional operation.

2.3 Internal PSTAR Communications

During operation the PSTAR must pass data between the pedestal, CIU, and transceiver box. Some of these digital signals have been decoded in work performed by David Giessel, a staff engineer at the Poker Flat Research Range. All of the information presented in this section is attributed David Giessel, and was not produced by the author, but are included for completeness.

The CIU and transceiver box communicate with one-way RS485 links. Messages have no apparent checksum and there is no "echo" after a message is received. Observations of the subsequent message from the pedestal suggests that all status parameters are embedded in the pedestal-to-CIU messages, therefore no echo is needed as the CIU only displays current pedestal status (e.g., if the CIU says "transmit on" and the pedestal misses that message, the CIU will not indicate that the transmitter is on). The CIU transmits messages to the pedestal on 1 Hz intervals. If no command messages are being sent to the pedestal, the CIU sends a "heartbeat" signal to the pedestal to verify that the link is still present on the same 1 Hz interval. The pedestal transmits its messages to the CIU at 15 Hz. These messages vary in length and payload as seen in Table 2.2 (RX messages).

Information from Robert Slye at NASA Ames indicates that when targets are present this message is 19 bytes long. The message format as given in his C code is shown in Table 2.3.

2.4 Beam Pattern

The beam pattern as described by the PSTAR documentation has a boresight gain of 19.5 dBi and a horizontal 3 dB beamwidth of 10.8°. S CW transmitter, keyed to a frequency near the PSTAR transmit frequency of 1.22 GHz, was set up within view of the PSTAR to experimentally corroborate this. The transmitter used was supplied and operated by Prof. Joseph Hawkins of the UAF Electrical and Computer Engineering (ECE) Department and was part of a parrot system being designed by Prof. Hawkins for use in producing simulated target reflections for the PSTAR.

With the transmitter on, the PSTAR was set to Passive mode and the I/Q data logged by the PXI ADC. Passive mode on the PSTAR is identical to Radiate mode with the exception of not transmitting a signal. As the PSTAR antenna swept through the location of the

Table 2.2. Logged messages to and from CIU.

RX/TX	Message Type	Length (Bytes)	Message
TX	CIU Heartbeat	2	0x7E00
TX	IFF Switch to Auto	4	0x7E80A0A0
TX	IFF Switch to Off	4	0x7E802020
RX	Ped Msg w/ Rad Off	7	0x7E201200290837
TX	Radiate On	4	0x7E804848
RX	Ped Msg w/ Rad On	7	0x7E201200290837
RX	Ped Msg w/ Rad On	6	0x7EC01200293B
TX	Ped Msg w/ Rad On	6	0x7EC09680FB12
TX	Set Mag 47.9	6	0x7EC09680FB92
TX	Set DLRP (init + 1)	20	0x7E8826184CEC8C4081F0CB084A6A8A000000003B
TX	Offset E +10001 N -30007	10	0x7EE0A600E488FF5193F7
RX	Ped Msg before Set Channel	7	0x7E201200110803
TX	Set Channel 1,2,3,4,5,19	7	0x7E200E2000F8C9
RX	Ped Msg after Set Channel	11	0x7E101200110E2000F808CE
TX	Disable Helo 1	4	0x7E80D0D0
TX	Enable Helo 1	4	0x7E805050

parrot transmitter the received signal strength varied proportional to the antenna beam pattern with a maximum at boresight. The attenuated sections of the data surrounding the transmit pulse were removed and the absolute value of the complex signal voltage was plotted versus time, which is related to azimuth. By visual inspection it was determined that the horizontal beam pattern was similar to a Gaussian function and so one was fit to the data as seen in Figure 2.9.

The result of the Gaussian fit was the function

$$f(a) = 0.48e^{-\left(\frac{a}{9.51}\right)^2}. \tag{2.1}$$

The maximum of the beam pattern function is 0.48 V at boresight (0°). A reduction in power of 3 dB can be achieved by dividing the voltage by $\sqrt{2}$, resulting in a half-power voltage of 0.34 V. The beam pattern function crosses the 3 dB boundary of 0.34 V at $\pm 5.3°$,

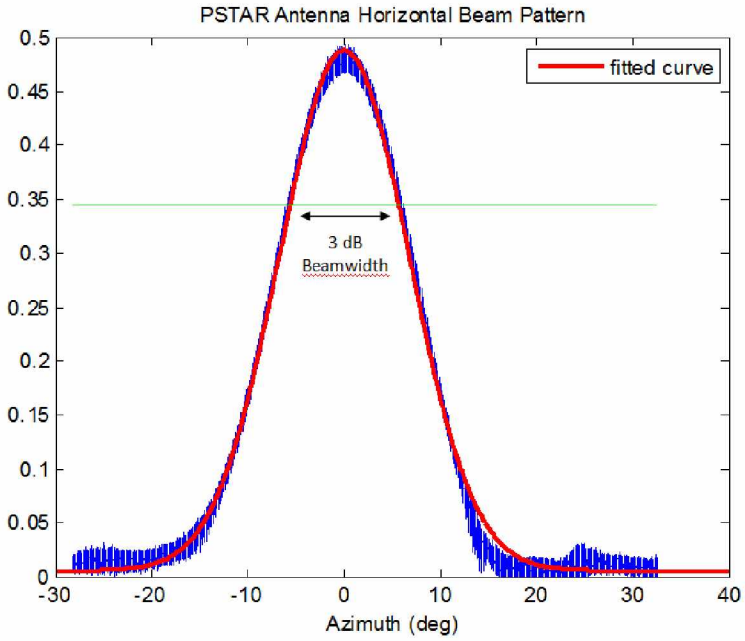

Figure 2.9. Horizontal beam pattern of PSTAR antenna.

resulting in a 3 dB beamwidth of 10.6°. This measurement is in close agreement with the PSTAR manual which lists the beamwidth as 10.8° [12]. The vertical beam pattern was not determined due to the difficulty of performing such a test.

Table 2.3. Target message format.

Byte	Name	Description
1	header	0x7E
2	count	
3	msgid	0xFE
4	timlsb	time stamp
5	timmsb	
6	ident	0x20 inbound
7	msgtype	0x3C target msg id
8	track	Target number
9	tent	Tentative hit (incl coast indicator bit)
10	quality	Signal Quality (incl fix bit)
11	east2	first 8 bits of Easting
12	east0	Last 4 bits of East first 4 of North
13	north1	Last 8 bits of Northing
14	ev	East velocity
15	nv	North velocity
16	me2	Unknown (east 8 bits of some sort)
17	me0	Unknown (east 4 bits and north 4 bits)
18	me0	Unknown (north 8 bits)
19	cksum	Checksum

Chapter 3
Estimation of Target Parameters

3.1 Radar Fundamentals

Before describing the details of obtaining elevation angle we will first discuss the basics of radar theory. The use of pulsed radio signals to range an object by reflections dates back to work by Briet and Tuve in 1926 [16]. Radar theory and design advanced rapidly during World War II and has continued to advance through current times [17]. While there are many types of radars in use today, the PSTAR is a monostatic pulsed system and that is the type investigated here.

3.1.1 Power

3.1.1.1 Radar Equation

One of the most fundamental ways to characterize a radar system is through the use of the radar equation. Different formulations of this equation exist for different types of radars, but for a monostatic system it is

$$P_r = \frac{P_t G^2 \sigma \lambda^2}{(4\pi)^3 R^4}. \tag{3.1}$$

This equation relates the power received (P_r) from a target reflection to characteristics of the radar and the target. The transmit power (P_t), antenna gain (G), and wavelength (λ) for the PSTAR are, respectively, 1 kW, 19.5 dBi, and 0.246 m, assuming a transmit frequency of 1.22 GHz. The variable σ represents the Radar Cross Section (RCS) of the target. This is a function of the size, shape, orientation, and composition of the target and generally must be determined experimentally for complex targets such as airplanes. The variable R is the range of the target from the radar. A target with a RCS of 1 m^2 will be assumed here as it is representative of the minimum RCS expected from a small single-engine aircraft such as the plane used as a cooperative target during tests. A plot of single pulse received power versus target range, starting at the minimum range of 1.2 km, is shown in Figure 3.1.

3.1.1.2 Noise

In addition to the power received from a reflected pulse the received signal will also contain noise. There are a large number of sources of noise that can be separated into two

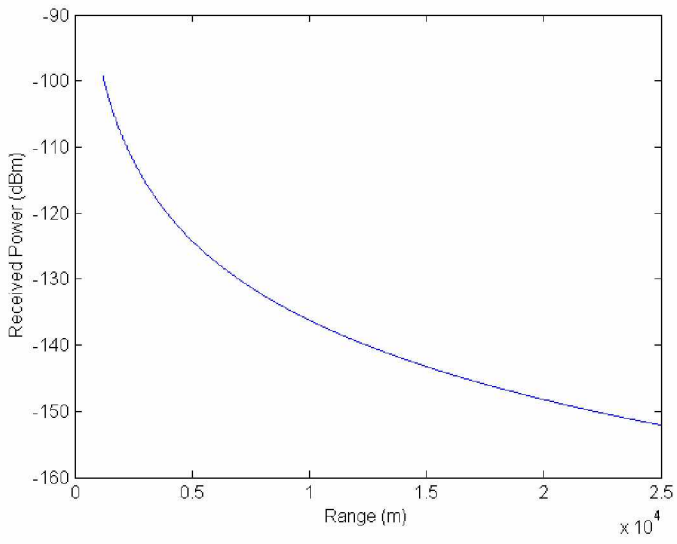

Figure 3.1. Power received versus range for a target with an RCS of 1 m^2.

groups: noise sources external to the radar and noise sources internal to the radar. The SNR of the transmit signal is very large and this signal will be attenuated significantly by the time it is received. As such, the noise contained within the transmit pulse can be ignored. The sources of noise external to the PSTAR are primarily related to the temperature of what the antenna is pointed at. The sources of noise internal to the PSTAR arise from the system temperature of the components along the receive path, and this is dominated by the noise figure of the first gain stage. While the details of the receiver hardware of the PSTAR are for the most part a mystery, the documentation does list a noise figure for the receiver of 2 dB. However, as the PSTARs used are not in mint condition it was important to determine the noise characteristics experimentally.

One method of determining the total noise is to determine the noise from all the major contributing sources and sum them to produce the total noise power. This can be useful for analyzing individual components within the receive path, but is not necessary for describing the system as a whole. Instead, the total noise power was measured by pointing the antenna towards the sky and sampling the receiver output. This data contains both sky noise and internal system noise and nullifies the need to know the receiver bandwidth. The one issue that did arise with the choice to measure noise in this manner comes from

the attenuation within the receiver during and immediately following the transmit pulse. Even when not transmitting the receiver still incorporates the attenuation. All samples within these periods of attenuation were ignored as done with the calibration data (see Section 5.2). The noise power, referenced to a resistance of 50 Ω, can be computed from the voltage data using

$$P_n = \frac{\overline{V^2}}{50}. \tag{3.2}$$

The result is a noise power of -42 dBm in the main channel and -34 dBm in the SLC channel at the baseband output of the receiver.

3.1.1.3 Signal to Noise Ratio

Up to now we have determined the received signal power at the input to the receiver and the total noise power at the output of the receiver. To determine the SNR of a received signal at the output of the receiver we must know the gain inside the receiver to relate the input signal power to the resulting output signal power. This was determined by injecting a signal of known power into the front end of the receiver and measuring the power of the resulting output signal. During calibration a -70 dBm signal is passed through a power divider delivering -73 dBm to both the Main and SLC channels. This signal passes through all cabling and the pedestal and so can be used to determine the gain of a signal received by the antenna. The maximum voltages of the resulting sinusoid on the Main and SLC channels, respectively, was 0.084 V and 0.147 V. These values equate to a power of -11.5 dBm on the main channel and -6.7 dBm on the SLC channel, referenced to an impedance of 50 Ω. These power levels indicate peak gain levels of 61.5 dB on the main channel and 66.3 dB on the SLC channel. The receiver gain can then be added to the expected received signal power from a 1 m^2 target (Figure 3.1) to produce an estimate of single pulse SNR for a target with a RCS of 1 m^2 (Figure 3.2).

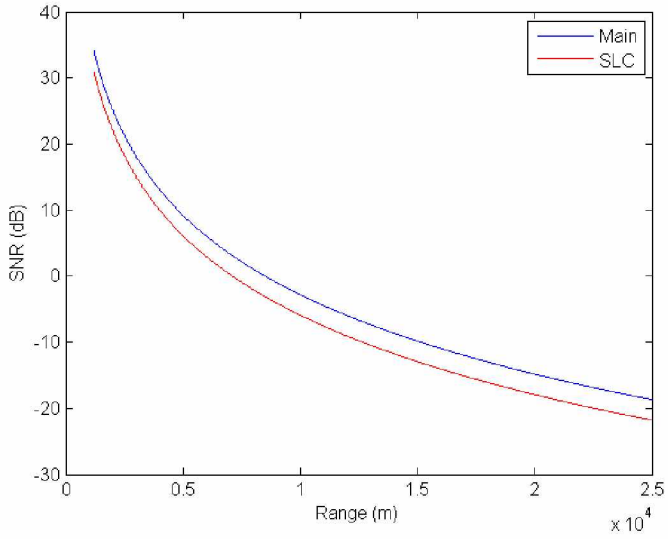

Figure 3.2. SNR versus range for a target with an RCS of 1 m^2.

This estimate is only for a single pulse, but in practice multiple pulses are combined to increase a target's SNR. When adding N consecutive pulses, the noise power is increased by a factor of N while the signal power is increased by a factor of N^2. The SNR resulting from the summation of N pulses containing coherent signals is shown in Equation 3.3,

$$SNR_{Total} = N_{Pulses} * SNR_1. \tag{3.3}$$

In addition to summing all the pulses of a single channel, the data from both channels can be combined to increase SNR further. The total SNR after summing over 128 pulses on both channels for a target with an RCS of 1 m^2 is shown in Figure 3.3.

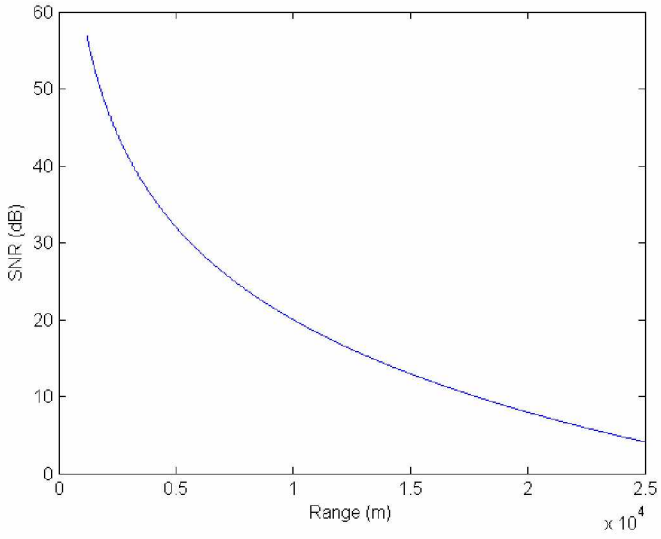

Figure 3.3. Total combined channel SNR for 128 pulses.

3.1.2 Azimuth

Unlike other target parameters, estimation of target azimuth does not require any computation for the PSTAR. Azimuth is estimated by noting the direction the antenna was facing at the time data containing a target signal was collected. Details of estimating azimuth on the modified PSTAR are discussed in Section 4.1.3. Other radar systems estimate target azimuth using different methods determined by the configuration of the antenna(s) used.

3.1.3 Range

A monostatic radar is simply a radar system in which the transmit and receive antennas are collocated. Generally, and in the case of the PSTAR, monostatic systems use the same antenna for both transmit and receiver, but this is not necessary. This differs from a bistatic, or multistatic, design which uses two or more antennas that are geographically separated. Pulsed radar systems operate by transmitting an RF pulse and computing the time delay between the transmission of a pulse and the reception of that pulse reflected off of a target. Many types of pulses can be used for a variety of reasons, but the simplest of them is a CW signal modulated with a rectangle function. This is the type of pulse transmitted by the

PSTAR system.

The received signal, ignoring frequency, will be a copy of the transmit signal scaled in amplitude and delayed in time. The time delay between transmission and reception of a pulse can be used to calculate the range of the target if the propagation velocity of the signal is known. In the case of the PSTAR, as in all radar systems, the propagation velocity is equal to the speed of light in the medium, or approximately $c = 3 * 10^8$ m/s. However, as the signal must travel twice the distance of the target to make a round trip the range can be computed as

$$R = \frac{ct_R}{2} \tag{3.4}$$

with the variable t_R representing the time delay between the transmission of a pulse and the reception of a target reflection. The range accuracy of a radar is dependent on the sampling rate of the system,

$$\Delta R = \frac{ct_s}{2}. \tag{3.5}$$

For the modified system the sampling rate, t_s, is 1 μs resulting in a range accuracy of 150 m. For a critically sampled system, where t_s = PulseWidth, the range resolution is defined by Equation 3.5. This defines the minimum distance targets must be spaced to avoid any overlap in received signals and is equal to 1.2 km for the PSTAR. Because the modified system is oversampled it is possible to detect two different targets with a smaller spatial offset, however the detection algorithm has not been optimized for detection of overlapping targets and no formal analysis of functional range resolution was performed. Any two targets spaced less than 1.2 km will result in one or more range gates containing power from both targets.

3.1.3.1 Range Ambiguity

While the above method of determining range works for a single pulse there is a complication when dealing with multiple pulses in succession, such as with the radar system described in this thesis. A radar's unambiguous range is defined by the IPP (Inter-Pulse Period), which is the inverse of the PRF. In the case of the PSTAR's 6.25 kHz PRF, the resulting IPP is 160 μs and the maximum unambiguous range is 24 km. Let us consider a burst of two pulses. After the first pulse, but before the second, any target return will obviously be

from the first pulse and the range can be correctly estimated using Equation 3.4. However, a target return detected after the second pulse could have originated from either pulse. The delay between the target signal and pulse that created it could be either X or X+IPP. The ambiguity in the time delay translates to an ambiguity in range and there would be two possible ranges for the detected target, separated by the maximum unambiguous range of the radar. For the system described here, ambiguity in range is not considered an issue worth mitigating. This is due to the fact that a target the size of an airplane, even a large one, would not be detectable past the maximum range of 24 km and hence would not be aliased to a closer range.

3.1.4 Doppler Velocity

The radial velocity of the target can also be determined from the received signal. If the target were stationary the received pulse would maintain the frequency of the transmit pulse, but a reflection off of a radially moving target will produce a frequency shift due to the Doppler effect. The Doppler effect essentially scales the transmit signal in time. For a target approaching the radar the received pulse will be slightly shorter than the transmitted pulse due to the target being slightly closer to the radar at the end of the transmit pulse than at the beginning. The received signal will contain as many periods as the transmit signal, but since the time over which the periods occur has decreased the frequency of the signal increases. The opposite is true for targets moving away from a radar. For these outward travelling targets the received pulse is slightly longer than the transmitted pulse causing a decrease in frequency. By comparing the frequency of the received pulse to the frequency of the transmitted pulse the radial velocity can be computed,

$$v_{dopp} = \frac{-cf_d}{2f_0}. \tag{3.6}$$

When working with the Doppler velocity it is important to remember that this is a measure of the velocity component towards the radar and will not be equal to the target's physical velocity unless the target is moving directly towards or away from the radar.

3.1.4.1 Doppler Velocity Ambiguity

As with range, there can be ambiguity in Doppler as well. The ambiguity in Doppler velocity is inversely proportional to the PRF. This is due to the PRF defining the sampling rate of a target's Doppler frequency. According to the Nyquist theory, to determine the frequency of a sinusoid the sampling rate must be at least twice the frequency of the sinusoid. If the Doppler frequency of a target is beyond the Nyquist limit for the sampling rate used (the PRF), then the actual Doppler frequency detected will be aliased to a lower frequency. In practice, Doppler ambiguity is not much of an issue for the PSTAR because it is unlikely that civilian aircraft will travel faster than the maximum unambiguous velocity, defined as

$$V_{max} = \pm\frac{\lambda(PRF)}{4}. \tag{3.7}$$

For a PSTAR operating with a transmit frequency of 1.22 GHz and a PRF of 6.25 kHz this equates to a maximum Doppler velocity of $\pm$384.4 m/s. This velocity is higher than can reasonably be expected for non-military aircraft. Additionally, since the Doppler velocity is the component of target's velocity radial to the radar it is normally less than the absolute velocity of the target. The Doppler accuracy of a system is defined by the number of pulses used in computing the Doppler spectrum,

$$\Delta V_{min} = \frac{2V_{max}}{N_{Pulses}}. \tag{3.8}$$

For the modified PSTAR this results in a Doppler accuracy of 6 and 3 m/s for a dwell time of 128 and 256 pulses, respectively. The Doppler resolution is a measure of how frequentially separate two different target returns from the same range must be to be detected as separate targets. As shown in Section 5.4.3, the Doppler spectrum of a radially moving point target is a delta function. This results in the Doppler resolution of the system being equal to the Doppler accuracy.

3.2 3D Target Location

As originally designed, the PSTAR system can produce azimuth, slant range, and Doppler velocity for detected targets. While this is enough information for some uses, it is not adequate here for reasons discussed in Chapter 1. The necessity of determining the three

(spatial) dimensional position of a target requires that elevation data is added to the range and azimuth data already determined by the PSTAR.

3.2.1 Triangulation

At the start of this project it was the intended to estimate target elevation angle through triangulation of range and azimuth data from multiple PSTARs. Each PSTAR has a known range and azimuth resolution and these can be used to generate a volume arc in which the target likely resides. The volume shown in Figure 3.4 is constructed using the PSTAR's range resolution of 1.5 km, azimuth resolution of 8°, and 3-dB vertical beam width of 28°. In practice, the PSTAR detects targets well above an elevation angle of 28°, further

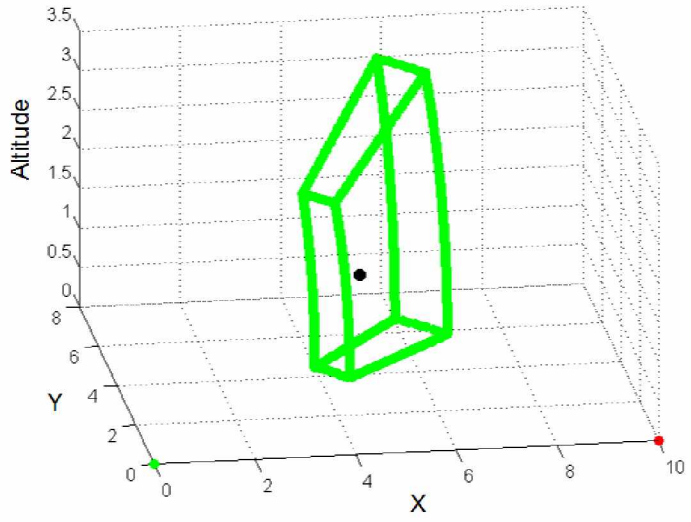

Figure 3.4. Volume containing target of detected by single PSTAR.

increasing ambiguity. However, as the 28° beamwidth will be used here as a cutoff for the sake of simplicity. The volume enclosing possible target locations can be decreased by incorporating data from multiple PSTARs and determining the overlap of volumes. This was simulated to determine the increase in accuracy gained by adding additional radars. An example of triangulation with three radars is shown in Figure 3.5.

It is clear in this example that while the volume has decreased through the use of mul-

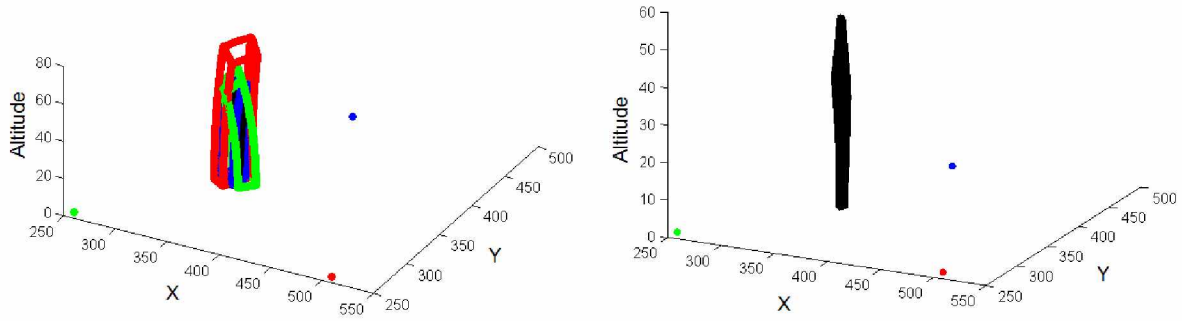

Figure 3.5. Target triangulation using three PSTARs. Left: Individual overlapping volumes. Right: Overlap region.

tiple radars the decrease is mainly due to a narrowing in the x-y axis leaving a still highly ambiguous z axis. Due to the geometry of individual radar target volumes being fairly vertical in extant adding additional radars to the system does little to decrease altitude ambiguity. The goal being to determine target altitude, the triangulation approach was shown to be insufficient.

In addition to providing poor altitude information, a triangulation based system would be much more complex than a single radar system. This would be due to the need to setup, power, and communicate with multiple spatially diverse radars. There would also be the need for the radars to operate on different channels to minimize interference which would result in a larger section of the wireless spectrum used. Even on different channels it would likely be necessary to control the rotation of the antennas such that no two antennas line up at boresight. The rotation of the individual antennas would also complicate target parameters because each radar will detect the target at a different time and location.

3.2.2 Interferometry

When the triangulation method originally suggested was ruled out as a possible approach for estimating altitude information spatial interferometry was investigated as an alternative. Spatial interferometry is a common technique in the radar world and requires the use of two or more receive antennas. If the antennas are spaced close enough together the phase difference between the received signals can be used to calculate the angle of arrival. A simple two antenna design is capable of resolving an angle in one plane and that is what

is investigated here (Figure 3.6).

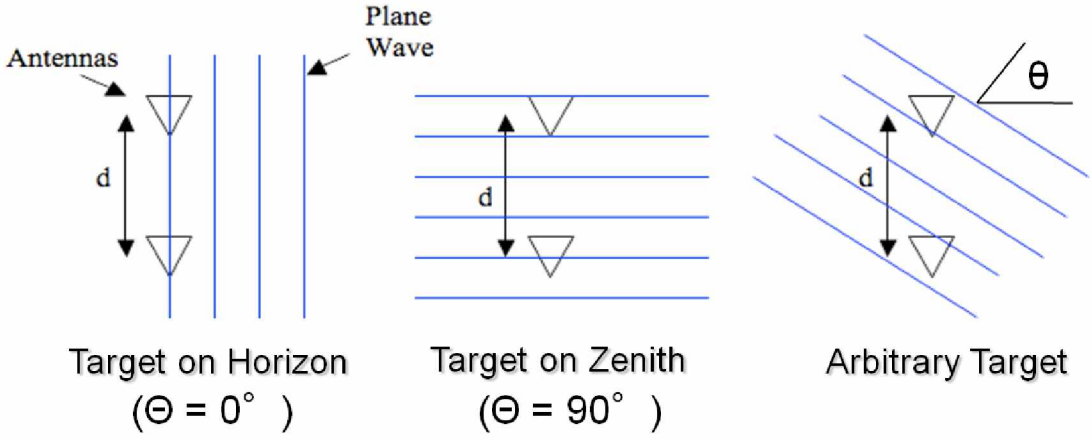

Figure 3.6. Spatial interferometer.

In the case of the 3-D PSTAR system elevation angle is needed, so two vertically mounted antennas where chosen. This vertical separation means that an incoming planar wave, such as a reflection from a target, will travel different distances to each of the two antennas depending on the angle of arrival. If the target is at the horizon, then there is no difference in distance to the antennas. If the target is at zenith, then the difference in distance will be at a maximum, that maximum being the physical spacing of the antennas. If the angle of arrival is somewhere between the horizon and zenith the difference in distance travelled will be physical spacing times the cosine of the angle of arrival. The difference in the physical distance travelled by the wave to reach the two antennas is determined by examining the phase difference between the two signals. With accurate phase measurements ($\Delta\phi$) the angle of arrival (θ) can be estimated using

$$\theta = \sin^{-1}\left(\frac{\Delta\phi}{2\pi}\frac{\lambda}{d}\right), \tag{3.9}$$

where λ is the carrier wavelength and d is the antenna separation, both in meters.

3.2.2.1 Elevation Ambiguity

Ambiguity of the angle of arrival occurs if the antennas are spaced more than a wavelength apart. This is due to multiple angles yielding the same phase difference. For example, a 30°

phase shift is indistinguishable from a 390° phase shift. This will become an issue for the PSTAR interferometer due to antenna size and frequency. The minimum separation possible is 48 cm. At the maximum operation frequency of the PSTAR (1.4 GHz) the wavelength is about 21 cm. This means that there will be ambiguity between angles of arrival. There are a couple methods to deal with this. The most straightforward being operating a lower frequency. Given the antenna spacing that we have available, we can have unambiguous elevation angles up to 30.8° for the first frequency channel (1.22 GHz).

One possible method for mitigating elevation angle ambiguity is to transmit multiple frequencies sequentially. For a given target, each frequency would produce a set of possible elevation angles. Because the 360° phase roll-over would occur at different elevation angles for the different frequencies the sets of ambiguous elevation angles would not be identical. The correct elevation angle would be the one present in all sets. This approach is more effective the further apart the transmit frequencies are spaced.

Chapter 4
Data Collection

For target information to be determined, the radar data must first be sampled and stored. This data includes azimuthal position from the pedestal, I/Q voltage data from the receiver, and timestamps from the computer logging the data.

4.1 Azimuth Data

The PSTAR operates with a rotating antenna, the position of which determines target azimuth. To match a detected target to a specific azimuth, the orientation of the antenna at the time in which the receiver data containing a target was sampled must be known. Multiple methods of obtaining azimuth data were investigated and are discussed within this section.

4.1.1 Digital Data from Pedestal

The first attempt at obtaining azimuth information was done by intercepting a digital signal sent from the Pedestal to the Transceiver box. This signal was located and decoded by PFRR engineer David Giessel. David produced a microcontroller based circuit that would wait for the RS485 message from the pedestal, decode it, and transmit a RS232 message containing the current azimuth to the serial port of the PXI system. This azimuth signal was sent every 64 ms, resulting in an azimuth being sent about every 4°. This signal was used to trigger sampling of the I/Q data resulting in data sets that were uniformly spaced in time. Due to a slight variation in rotation speed, uniform sampling in time does not result in uniform sampling in azimuth. Over multiple rotations the azimuths sampled drift slightly so that data may be collected at an azimuth 10° on one pass, and then at an azimuth of 11° on the next. For a portion of the development period it was intended to use a cluttermap to mitigate the effects of stationary ground clutter. The use of a cluttermap required that the same azimuths were sampled on each rotation of the antenna. The need for data sampled at repeatable azimuths forced the abandonment of this technique for obtaining azimuth position. Since a cluttermap is not used in the final implementation, it is likely that the method for obtaining azimuth described above would be adequate, however, this was not verified. Details on the cluttermap can be found in Section 7.1.2.

4.1.2 Resolver

The PSTAR antenna is attached to a large gear that is connected to an electric motor that rotates the antenna. Also attached to that large gear is a Harowe 11BRCX-300-T95E brushless resolver. Resolvers are rotary transformers with three windings. One is an exciter winding into which a reference AC signal is fed. The two remaining windings produce outputs that are functions of the shaft angle and are 90° out of phase. By comparing the relative phase of the two windings, the angle of the shaft can be determined as well as the rotational velocity [18].

The reference signal produced by the PSTAR pedestal was measured to be a 2.9 kHz sinusoid with a RMS voltage of 8 V. A Control Sciences, Inc. 268D306 Resolver-to-Digital Converter (RDC) was purchased to convert the analog resolver signals to a parallel digital signal to be read by the PXI system via a National Instruments PXI-6509 Digital Input-Output (DIO) card. Position signal from the RDC was successfully captured by the DIO card and converted into a numerical value on the PXI system. After getting the resolver set up properly it was discovered that a full cycle of the resolver consists of about 10 degrees of antenna rotation due to a combination of the gear ratio of connecting the resolver to the antenna mount and the resolver being a four-pole resolver. After some investigation it was determined that the azimuth position signal rolling over 36 times per rotation produced an ambiguity problem too difficult to overcome, so the resolver was abandoned. It is speculated that the PSTAR does not use the resolver for angular position measurements, but rather, uses it for angular velocity measurements to feed back into the pedestal motor controller.

4.1.3 Hall Effect Sensor

The method of estimating azimuth used in the final system is described in this section. During investigation of the resolver it was noticed that a small sensor was located underneath the outer edge of the antenna gear. Attached to the underside of the antenna gear is a small magnet that passes directly over the sensor as the gear rotates. The movement of the magnet over the sensor produces a square-wave on the sensor output. The result of this is that as the antenna rotates a short pulse is generated once per revolution. Monitoring

this signal gives a periodic indication of absolute position and can be used for determining azimuth. The time between pulses is monitored and this provides an accurate measure of rotational velocity, which can be used to calculate the time delay between specific azimuths. For example, if it was desired to sample every 5° and a rotational period was 6 s, then sampling should occur at the time of the zero crossing and every 83.3 ms to sample all azimuths from 0-355° at 5° increments. A connector was assembled to allow easy access to the internal pedestal wire that caries the Hall effect sensor signal. Installation requires the temporary removal of the rectangular box mounted to the side of the pedestal, an existing internal connection disconnected, and the new connector installed in-line with the old connection allowing for access to the Hall effect signal without disrupting any PSTAR functionality. The PSTAR pedestal is not modified in any way other than the pass-through connector so it can be restored to original condition simply by removing the connector. It should be noted that this method is only accurate if the rotational velocity of the antenna is constant. Under high wind loads the rotational velocity may be variable, which would induce error in the azimuth estimations. Monitoring rotational velocity with the resolver would be a means to mitigate this error.

4.2 I/Q Data

The data containing target information comes from the output of the PSTAR receiver. This output is in the form of two pairs of baseband I/Q channels, one pair for each antenna. The times at which the I/Q data is sampled is necessarily constrained by the relative times of azimuths as well as transmit pulses because a target's range is a function of the delay after the transmit pulse and the azimuth is a function of the delay after the pedestal zero crossing.

4.2.1 Analog to Digital Converter

The signal output from the receiver is an analog signal, and so must first be digitized before it can be stored. The digitization is performed by an Analog to Digital Converter (ADC). The first ADC investigated was the ADLINK PCI-9812, a PCI based ADC card that is installed in a standard desktop PC. This card was eventually abandoned after discovering that it was incapable of triggering data collection in the necessary manner. At a specified

azimuth the ADC must sample in bursts after successive transmit pulses for an arbitrary duration of pulses. The PCI-9812 was capable of triggering a burst of data collection after the transmit pulse, but then the card had to be reinitialized before it could trigger off another pulse. The delay between the cessation of a burst of data collection after one pulse and the resumption of data collection after the following pulse, 10 μs for the 6.25 kHz PRF, proved too short for the card to reinitialize in time to trigger after the following pulses. The algorithm used for computing Doppler velocity requires that the sampling bursts must be uniformly spaced in time forcing the need to burst sample after successive transmit pulses without missing any.

After determining that the PCI-9812 card was not suitable, the decision was made to use the ADLINK PXI-9816D/512(G) ADC card. This card is capable of re-triggering on successive transmit pulses without the need to change any register settings on the card between pulses. The move to the PXI-9816 necessitated the move to a PXI based PC to accommodate the card.

4.2.2 Dual Antenna

As needed by the spatial interferometry technique used to determine elevation angle, two antennas must be used for receiving target echos. A dual antenna was created by combining two PSTAR antennas into a single enclosure. The original PSTAR antenna consists of five panels of dielectric material with the elements of the antenna fixed to them (Figure 4.1).

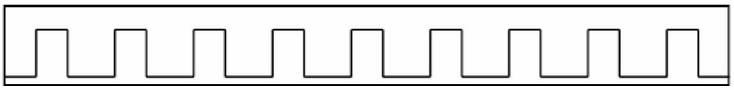

Figure 4.1. Linear array of antenna elements on dielectric panel.

The lower four of these panels are identical and make up the main antenna. The top panel is slightly shorter and has dipoles fixed to the top that protrude perpendicular to the plane. This is the antenna for the Interrogate Friend or Foe (IFF) system. Since the IFF system is not used in this application, the IFF antenna was not included. Each of the four needed panels are mounted at the base to a metal back-plane. There is a spacing of 11 cm between panels and 7.5 cm at the top and bottom (Figure 4.2).

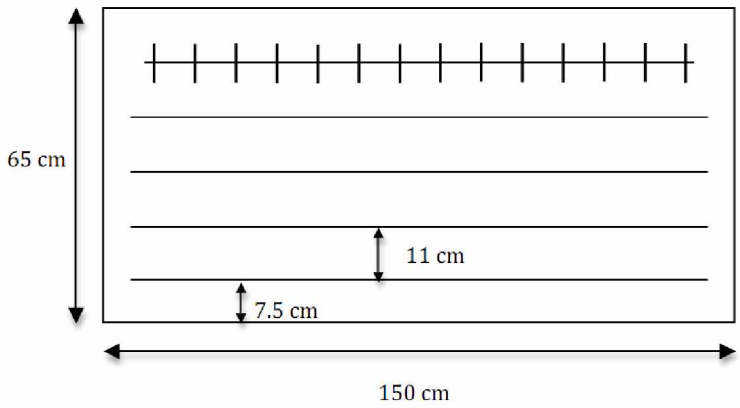

Figure 4.2. Arrangement of panels in original PSTAR antenna.

The lower four panels, along with the attached back-plane, were removed from the original two antennas. These sections from the two antennas were joined to produce one larger structure that contains two antennas (Figure 4.3).

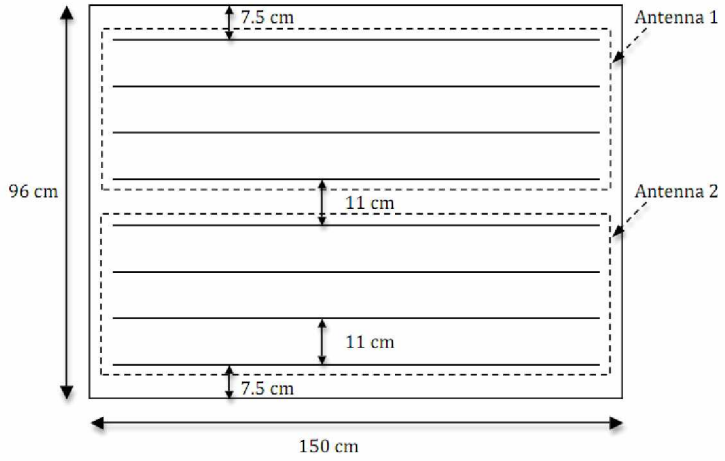

Figure 4.3. Modified antenna design.

This structure was then installed in a custom-made fiberglass enclosure. The enclosure is made of two symmetric pieces (front and back) that are joined around the edges with an aluminum band. The RF connections for the two antennas within the enclosure are on the back side of the antenna, one being at the bottom, and the other about half way up.

The fitting to mount the antenna onto the PSTAR pedestal was removed from one of the original antennas and fitted to the bottom of the dual antenna. All design and fabrication of the dual antenna housing was performed by then UAF Mechanical Engineering undergraduate student Matthew Van Atta. A comparison of the dual antenna and the standard PSTAR antenna can be seen in Figure 4.4.

Figure 4.4. The dual antenna (back) and original PSTAR antenna (front).

4.2.3 PXI System

The choice of a PXI based ADC card dictated the use of a PXI based PC to facilitate data collection. The PXI chassis selected was the ADLINK PXIS-2558T. This chassis was selected mainly due to the inclusion of a touch screen which facilitates easy use in the field. The PXI chassis contains a PXI back-plane into which PXI cards can be added and at a minimum a controller card must be installed in the back-plane. A PXI controller card is essentially an entire PC and controls the rest of the PXI system. The controller chosen for use was the ADLINK PXI-3950. This card was chosen primarily for the dual core processor included

because current versions of LabVIEW software include optimizations for utilizing multiple processor cores. As precise timing is necessary within the data collection program it was desired to ensure adequate system resources are always available to LabVIEW since Windows XP is certainly not a real-time operating system.

The zero crossing signal from the pedestal is a digital signal and required the inclusion of a digital input card into the PXI system. The card selected was the National Instruments PXI-6509 as it had already been purchased for interfacing with the RDC. As stated before, the ADC card chosen to digitize the I/Q signals was the ADLINK PXI-9816D/512(G). The final PXI component included was the National Instruments PXI-5651 RF Signal Generator card. This card is used to generate a signal used in the calibration routine. An image of the PXI system is shown in Figure 4.5.

Figure 4.5. PXI system as used in 3D PSTAR system.

4.3 Modified PSTAR System

All components of the 3D PSTAR described above must operate together to gather data correctly. The modified antenna mounts to the PSTAR pedestal in the same manner as the original, through the use of a metal collar. The bottom antenna is connected to the main channel type-N connector (J7) while the top antenna is connected to one of the IFF TNC connectors (J8). These two signals pass through a rotary coupler inside the pedestal and exit through stationary connectors on the bottom of the pedestal. The main channel on the bottom of the pedestal (J4) is connected to the main channel on the transceiver box (J4), while the the IFF connection on the bottom of the pedestal (J5) is connected to

the SLC A input located at the bottom of the transceiver box (J5). The signal from the Hall effect sensor is tapped inside the pedestal on pin S of the connector joining the side-mounted electronics module to the main pedestal assembly. A T-connector was created to facilitate access to the signal without disrupting PSTAR operations. The tapped signal line is connected to PXI-6509 connector block, which itself is connected to the PXI-6509 DIO card.

The ADC card is connected to the receiver I/Q channels through the PSTAR 3W1 wiring harness located inside the back door of the transceiver box. In normal operation, the I/Q channels are connected to the PSTAR processor subsystem (A1P2) with connector P8 of the 3W1 wiring harness. In the modified system the I/Q signals are not connected to the PSTAR processor and instead the P8 connector was removed and BNC connections added to each of the individual coax lines contained within the connector. The I/Q signals for both the main and SLC A channels are then connected to the PXI ADC card through four cables of equal length. The Main I, Main Q, SLC A I, and SLC A Q signals connect to the A0, A1, A2, and A3 channels on the ADC card, respectively. The transmit pulse envelope signal is used to trigger data collection and is accessed by a connection made between pin B on connector J1 of the Power Amplifier (PA) module to the trigger input on the PXI ADC card. The PSTAR is powered by an Iota Engineering DLS-27-40 power supply. This power supply delivers 27.2 VDC at a maximum of 40 A to the PSTAR. The connection is made through the power connection on the front of the PSTAR (J1) and utilizes the PSTAR power cable (W11) with one end modified to attach to the terminals on the power supply. While in Radiate mode the PSTAR draws around 25 A from the power supply. A wiring diagram for the operational mode is shown in Figure 4.6.

The PXI RF signal generator card is left unconnected during normal operation, but must be connected during the calibration routine. During calibration the signal generator output is connected to a 3 dB power divider. The two cables normally connected to the antenna are connected to the power divider during the calibration routine. It is very important to ensure that the PSTAR is never put into radiate mode while in the calibration configuration because the transmit pulse would likely damage the PXI signal generator card and perhaps other components of the PXI system. A wiring diagram for the calibration mode is shown in Figure 4.7.

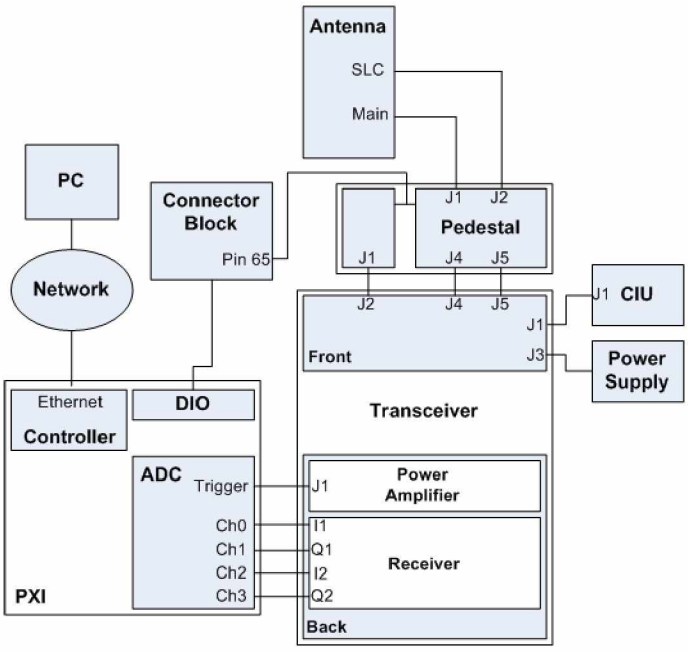

Figure 4.6. Operational 3D PSTAR wiring diagram.

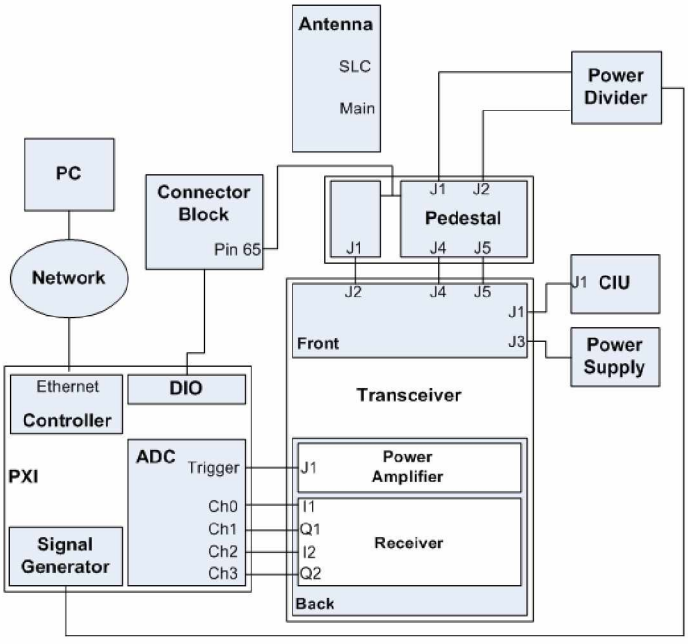

Figure 4.7. Calibration PSTAR wiring diagram.

4.4 Data Collection Program

4.4.1 North Alignment

Obtaining a valid north alignment of the system is necessary for the measured azimuths of targets to be accurate. The alignment is accomplished by probing the Hall effect signal with a voltmeter while manually rotating the antenna. The voltage should read zero for everywhere other than directly at the zero crossing (magnet location) and 5 V when the antenna is aligned directly at the zero crossing. The rising edge of the pulse created by the Hall effect sensor is used as the azimuth reference, so it is desired to align the antenna right at the azimuth where the voltage jumps to 5 V. The angle between this azimuth and true north was measured using the known bearing to a geographical marker. In practice, it proved difficult to achieve a precise north alignment through this method and the datasets collected were manually rotated to the correct azimuths using cooperative target GPS data. Once the north alignment has been completed it will stay valid until the PSTAR is physically moved. Improvements to the reliability of the north alignment routine are discussed in Section 7.2.

4.4.2 I/Q Data

There are two types of I/Q data collected during operation of the PSTAR, calibration and operational data. Both of these datasets are stored for later post-processing. The LabVIEW code for the data collection program can be found in Section A.1.

4.4.2.1 Calibration Data

The purpose of the calibration routine is to determine the relative phase difference between the two channels, which is necessary in the determination of target elevation angle. Before collecting calibration data it is important to allow the PSTAR to warm up for at least 15 minutes. It is only necessary for the calibration to be performed once per operation of the PSTAR, although it is useful to collect two calibration datasets because occasionally one of the datasets will not be valid.

The calibration is performed, after sufficient warm-up, by setting the PSTAR to STANDBY mode on the CIU and connecting the PXI signal generator to the PSTAR as shown in Fig-

ure 4.7. The start-up calibration LabVIEW program (*Startup_Cal.vi*) is then initiated on the PXI system. Note that the first step of the program is to present the user with a dialog box warning not to set the PSTAR to RADIATE, as doing so may cause damage to the PXI system. This program initiates the creation of a 1.2200002 GHz sinusoid at a power level of -70 dBm by the PXI signal generator. This signal passes through both the Main and SLC A channels of the receiver and the resulting four baseband I/Q signals are sampled at 1 Msps for 20,000 samples. This data is then stored in an ASCII file labelled as *Cal_YY_MM_DD_HH_MM_SS.dat* indicating the time and date at which the calibration file was created.

4.4.2.2 Operational Data

The operational data is the I/Q data collected while the PSTAR is actively running. The data is collected by first connecting the PSTAR as shown in Figure 4.6 and allowing the PSTAR to warm-up up for at least 15 minutes. The PSTAR is then set to RADIATE on the CIU and the LabVIEW data collection program is run. Two data collection programs can be chosen from, *Record_Data_10deg_256pt.vi* and *Record_Data_5deg_128pt.vi*. The difference between the two programs is in how many azimuths are sampled per revolution and how many pulses are sampled per azimuth. One program samples azimuths in 10° spacings, resulting in 36 azimuths per revolution, with 256 pulses sampled per azimuth. The other samples azimuths in 5° spacings, resulting in 72 azimuths per revolution, with 128 pulses sampled per azimuth. Other than the azimuth spacing and the number of pulses collected per azimuth, the two data collection programs are identical.

A simplified flowchart describing the data collection process can be seen in Figure 4.8 while the complete LabVIEW code can be found in Section A.1.

The data collection program functions by first allowing the pedestal to rotate until a zero crossing occurs, as indicated by the Hall effect sensor. The time at which this occurs is noted and the program waits for another zero crossing signal. The times of the two zero crossings are used to compute the time per revolution. The PSTAR documentation states the time per revolution to be 6 s, which is very close to the observed time of about 5980 ms. The measured time per revolution is used to compute the time delay between azimuths. For example, if the program is to sample at 10° increments, 72 increments per

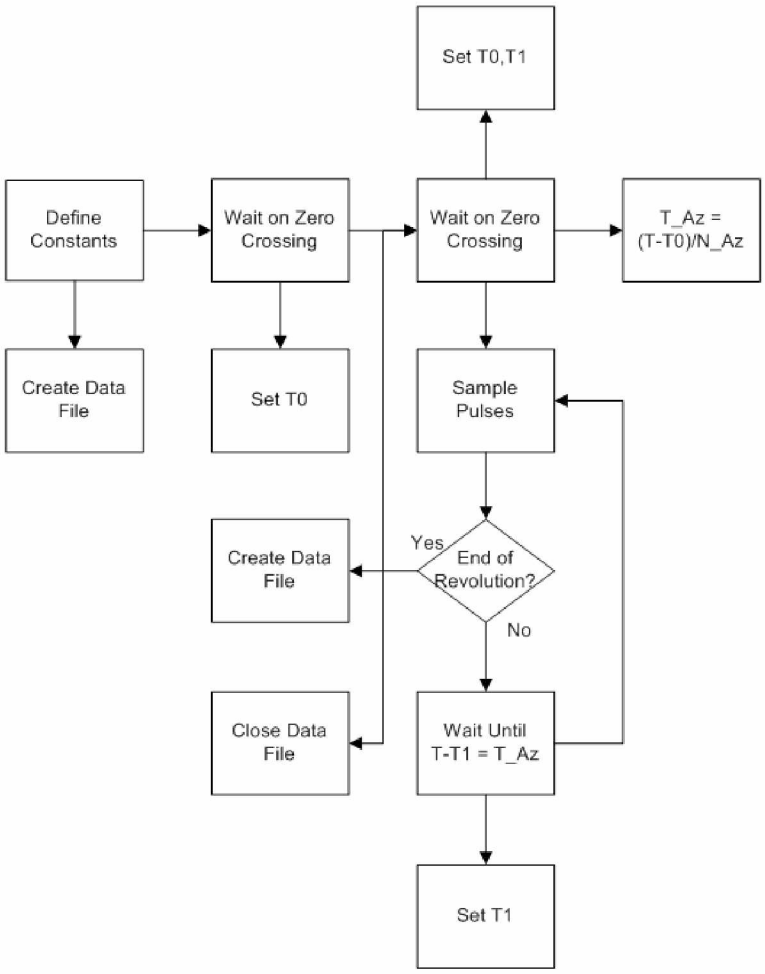

Figure 4.8. Data collection program flowchart.

revolution, and a complete revolution occurs in 5980 s, then the wait between azimuths is: 5980/72 = 83 ms. Once this has been computed the program waits for the next zero crossing. At this time the first group of pulses is immediately sampled and written to a .tdms binary file. Once the data sampling is complete the program determines how much time now remains until the next azimuth will occur. For the 5° program this is usually about 20 ms, but it varies due to the non-deterministic timing nature of running software on Windows XP. The program waits for the required amount of time and then repeats the sampling process for the next azimuth. This process repeats until a full revolution has been completed. All data collected in the revolution is stored to the same data file which

is closed upon completion of a revolution and a new file is created for the next revolution. A limitation of the software is that the opening and closing of data files at the end of a revolution cannot occur fast enough to catch the zero crossing immediately following sampling the final azimuth, resulting in the sampling of every other rotation. The time at which the next zero crossing is observed is actually two revolutions past the previous observed zero crossing. The time per revolution used to determine azimuth time spacing is updated from the new time between zero crossings, but that time must be divided by two to account for the two rotations. The calculation of the updated azimuth time spacing occurs concurrently with the collection of data at the zero crossing and is completed by the time the sampling is complete and the wait time is needed.

The collected data is stored in .tdms files with one file created per revolution. The Technical Data Management Streaming (TDMS) file format is a structured binary format developed by National Instruments designed to be highly organized while keeping file size to a minimum and allowing streaming access. The files created by the data collection program are titled as *YY_MM_DD_HH_MM_SS.tdms* indicating the time and date at which the file was created. The data inside is divided into channel groups with one channel group per azimuth and the name of each channel group being the azimuth it contains. For example, if the program is sampling 72 azimuths per revolution and the zero crossing on the pedestal is pointed directly north the TDMS file will have 72 channel groups titled 0, 5, 10, ..., 355. If north was 10° past the zero crossing, then the resulting channel groups would be title 350, 355, 0, 5, ..., 345 with the channel group titled 0 indicating the north azimuth. Each of these channel groups contains a timestamp indicating the time at which sampling of that azimuth began. Inside each channel group are four channels labelled I1, Q1, I2, and Q2 which contain voltage data for each of the receiver channels. Each of these channels contains all the samples from all the pulses collected on that channel at that azimuth. The sampling of each pulse is triggered by the falling edge of the 8 μs transmit pulse which initiates a burst of 150 samples at 1 Msps. As data from all pulses per dwell are contained within a channel, the number of samples contained within that channel will be 150 times the number of pulses per dwell, either 128 or 256 depending on the data collection program being ran. This results in either 19,200 or 38,400 samples per channel. A diagram showing the structure of a *.tdms data file is shown in Figure 4.9.

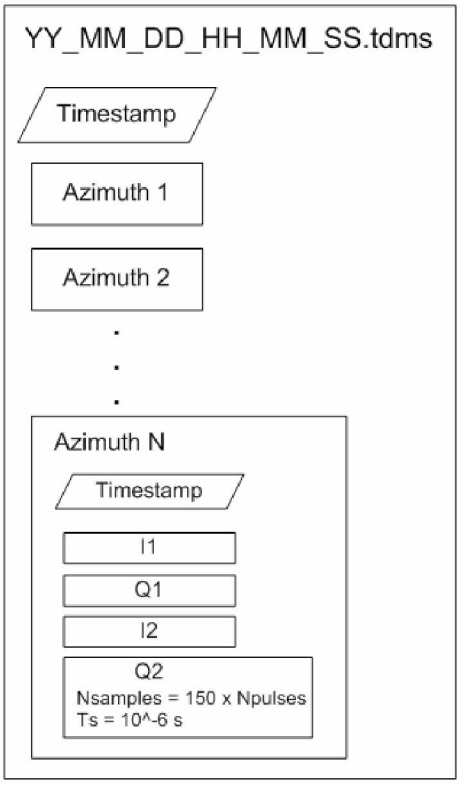

Figure 4.9. TDMS file structure.

Chapter 5
Data Processing

5.1 Overview

This section discusses the details of processing radar data using functions written in MATLAB. The data collected during operation must be post-processed to determine the possible presence of targets. The processing of a batch of data requires a single **.dat* ASCII calibration data file and any number of **.tdms* operational data files. A flowchart describing the data processing algorithm is shown in Figure 5.1.

When running the MATLAB data processing program (*Run_All_Folder_3D.m*) it directs the user to select the calibration file and a folder containing the operational data. All **.tdms* files in the selected operational data folder are processed and all files of other formats are ignored. It is important to only have **.tdms* files produced by the data collection program within this folder. While the data is being processed two status bars are displayed to the user indicating the status of processing the current file as well as the overall status of the batch. After the data processing program has terminated a variable called **Targetsf** will be present in the MATLAB workspace. **Targetsf** is a Nx7 array where N is the number of targets detected in the dataset. The columns of the array, from first to last, are Azimuth (°), Range (m), Doppler Velocity (m/s), Elevation Angle (°), Hour, Minute, and Second. The time stored in the last three columns is the time at which the start of sampling at that azimuth began. In addition to the numerical target information contained within **Targetsf** a radial plot of the targets is displayed. This plot indicates targets with an X and contains 5 km range rings. The time of the detection, relative to the times of other target detections within the dataset is indicated by the color of the X with blue indicating the oldest detections and red indicating the newest detections. An example of the data processing program's output can be seen in Figure 5.2 and Table 5.1. Figure 5.2 contains data taken over a period of about 5 min and shows a plane approaching the radar from the west. The same target is often seen in adjacent azimuths causing multiple target hits per sweep, e.g. the three purple X's near the 15 km range ring are from the same physical target. Table 5.1 shows the first 10 targets detected within the dataset. The full dataset contains 46 target hits. The MATLAB code for the data processing can be found in Section A.2.

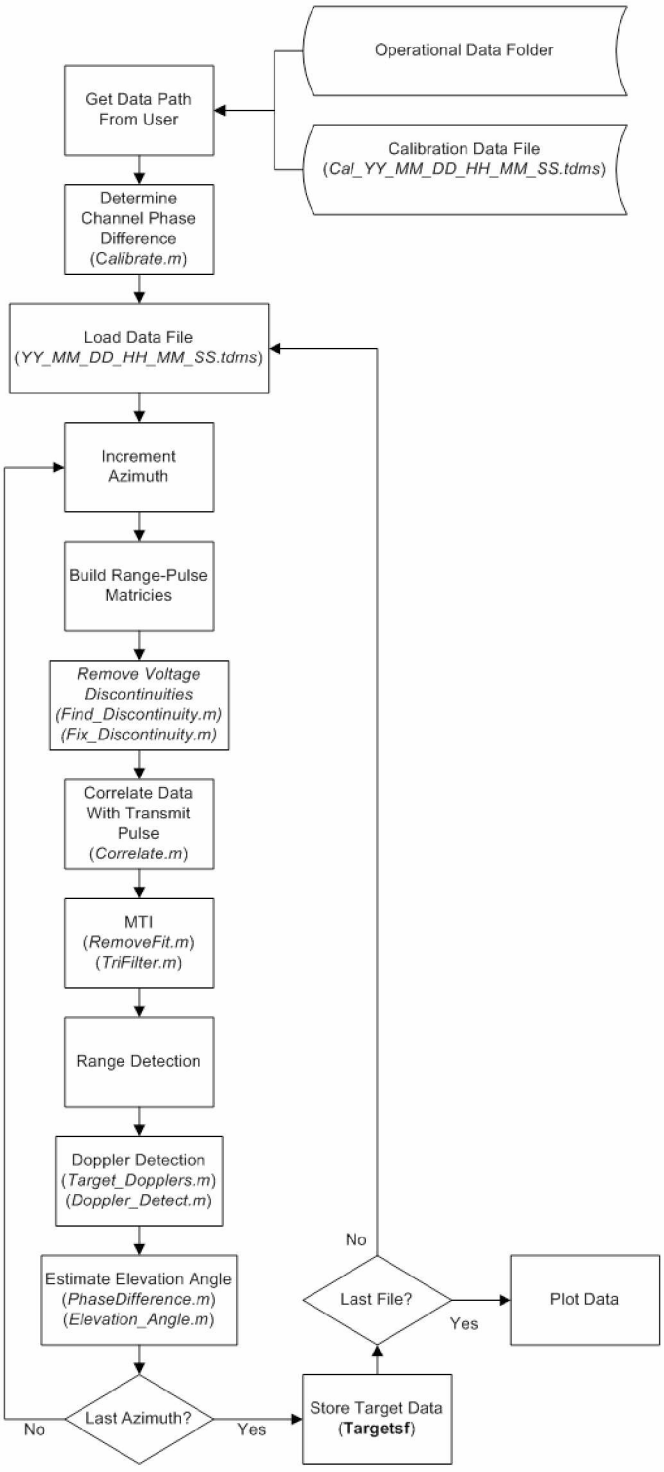

Figure 5.1. Flowchart of data processing algorithm.

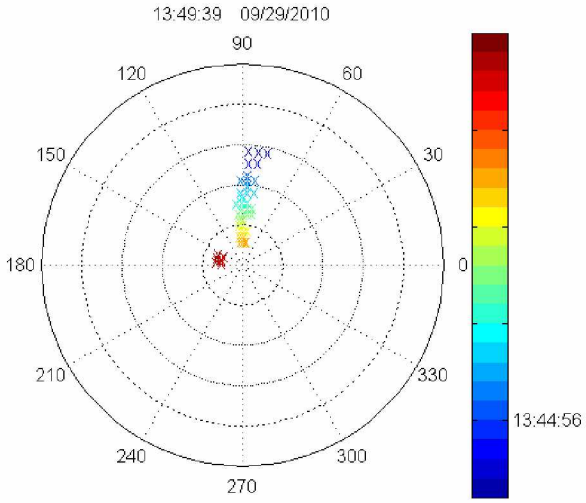

Figure 5.2. Example of data processing plot output.

Table 5.1. Example of data processing numerical output.

Az (°)	Range (m)	Dopp Vel (m/s)	Elev Ang (°)	Hour	Min	Sec
79	14250	-69.6	20.3	16	45	16.67
84	14250	-69.6	20.2	16	45	16.75
89	14250	-69.6	21.3	16	45	16.83
84	12750	-63.5	30.3	16	45	40.58
89	12750	-51.4	20.8	16	45	40.65
89	11250	-63.5	23.1	16	46	4.53
84	10650	-63.5	20.23	16	46	16.39
89	10650	-63.5	56.54	16	46	16.47
94	10650	-63.5	55.14	16	46	16.56
89	9900	-63.5	22.84	16	46	28.40

At the initialization of the data processing program multiple constants are defined. These constants can be separated into three groups: always constant, defined during data collection, and defined during data processing. The always constant constants will be

the same for all datasets collected using the current hardware and software. The defined during data collection constants are variables that are dependent on the dataset being used and must be changed to correspond correctly to the dataset being processed. The defined during data processing constants are independent of the dataset being used, but alter how the data is processed. A summary of the data processing constants is listed in Table 5.2.

Table 5.2. Data processing constants.

Always Constant

	MATLAB Variable	Value
Pulse Width	**tau**	8 μs
Sampling Time	**ts**	1 μs
Samples per Pulse	**Nsamples**	150
Antenna Spacing	**d**	0.48 m

Defined During Data Collection

	MATLAB Variable	Value
Carrier Frequency	**Fc**	1.22-1.4 GHz
PRF	**PRF**	5.55 or 6.25 kHz
Pulses per Dwell	**Npulses**	128 or 256

Defined During Data Processing

	MATLAB Variable	Value
Minimum Doppler Velocity	**Min_Dopp**	25 m/s
Discontinuity Threshold	**DiscThresh**	1.2
Polynomial Fit Degree	**PolyDeg**	2
Point Target Filter Length	**TriFiltLength**	5
Point Target Filter Weight	**TriFiltWeight**	0.9
Doppler Threshold	**DoppThresh**	10

5.2 Calibration

The first step of the data processing program is to determine the phase difference between the two channels. This is accomplished using the data stored in the calibration **.dat* file.

The data from the file is loaded into four arrays: **I1**, **Q1**, **I2**, and **Q2**. Each of these arrays consists of a sinusoid that has been continuously sampled at 1 Msps for 20,000 points. As discussed in Section 2.1 the PSTAR receiver is highly attenuated during the transmit pulse and the attenuation is decreased incrementally to none at a range of 6 km. As a first step to processing the calibration data all data within the periods of attenuation are removed. A comparison between the data before and after removal of the attenuation steps can be seen in Figure 5.3, note that only a portion of the entire dataset is shown here.

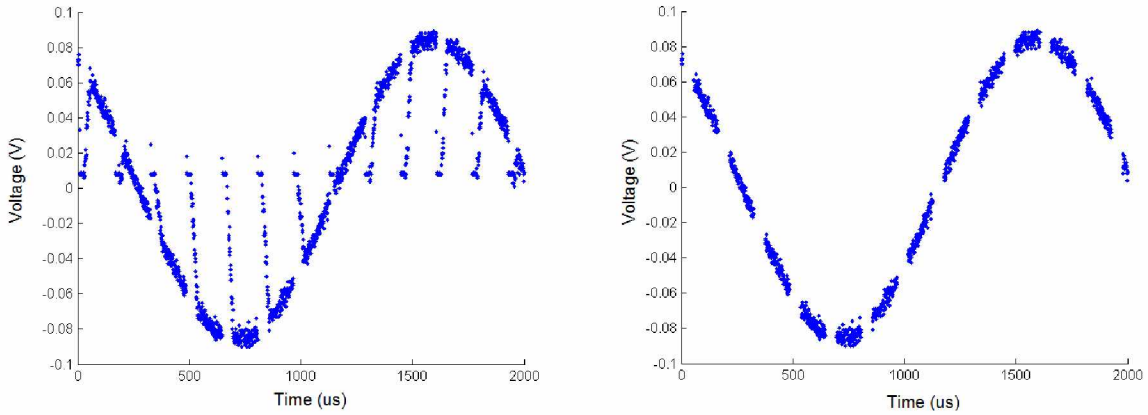

Figure 5.3. Removal of attenuated samples from calibration data. Left: Unmodified data. Right: Data after attenuated samples are removed.

Next, DC offsets are removed from the data. The mean of each array is then subtracted from every point within the arrays to center the waveform about 0 V. The data is now in the form of a noisy sinusoid and parameters of the sinusoids are found by fitting the function

$$f(t) = A\sin(\omega t + \phi) \tag{5.1}$$

to each channel's data. The best fits for the three variables A, ω, and ϕ are computed using MATLAB's *fit()* function (Figure 5.4). The ϕ variable corresponds to the phase of the sinusoid at the beginning of the sampling period. The phase difference between channels is determined by comparing the ϕ of the Main channel (**I1**, **Q1**) to that of the SLC A channel (**I2**, **Q2**). The value used as the channel phase difference is the average of the phase difference between I and Q channels,

$$\Delta\phi = \frac{(\phi_{I1} - \phi_{I2}) + (\phi_{Q1} - \phi_{Q2})}{2}. \tag{5.2}$$

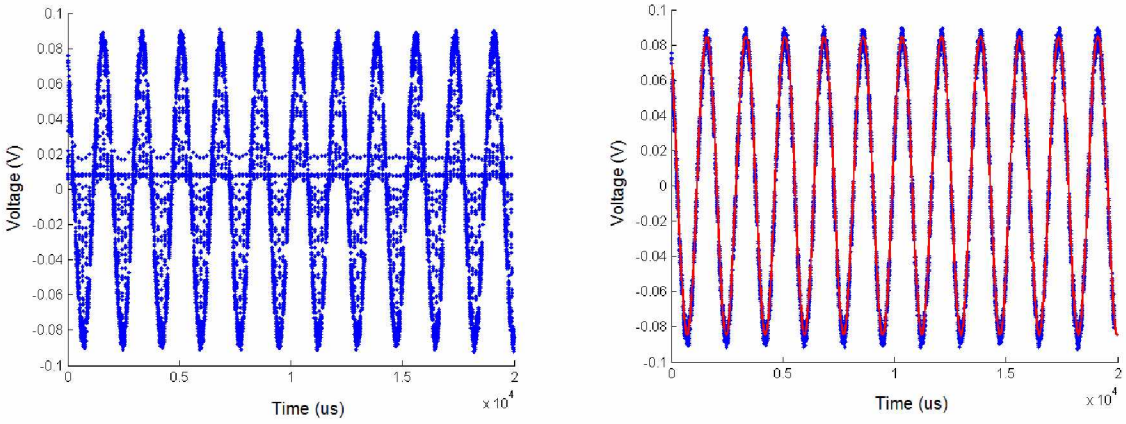

Figure 5.4. Raw and processed calibration data. Left: Unmodified calibration data. Right: Processed calibration data (blue) with fitted curve (red).

5.3 Signal Conditioning

5.3.1 Data Packaging

The data processing program loads only one operational data file at a time, corresponding to one complete revolution, starting with the first listed in the directory being processed. Due to the chronological naming of the data files the first listed in the directory is the first that was recorded. The **.tdms* data file is loaded into MATLAB with the use of the function *convertTDMS.m* v1.7. The function, originally created (v1.0) by Brad Humphreys of ZIN Technologies on 4/23/2008 and then edited by six others to reach v1.7, was downloaded from the MATLAB Central File Exchange [19]. The data contained within the **.tdms* file is loaded into six MATLAB variables: **I1a**, **Q1a**, **I2a**, **Q2a**, **Az**, and **Time**. The first four contain the voltage data sampled from the corresponding channels in a MxN array where M is the number of points recorded per azimuth and N is the number of azimuths per revolution. For example, data sampled at 5° increments would result in an array of size 19200x72. The **Az** variable is a Nx2 array containing azimuth angles in the first column and the corresponding time at which the sampling of that azimuth occurred in the second column. The **Time** variable contains a string indicating the time at which the sweep contained in the dataset began. This time is the same as that contained within the data filename.

The data processing program now loops through all azimuths. At the start of each

azimuth loop iteration the data contained in the column corresponding to the current azimuth of **I1a**, **Q1a**, **I2a**, and **Q2a** is restructured into four new arrays: **I1**, **Q1**, **I2**, and **Q2**. Each of these is a Px150 array where P is the number of pulses per dwell, either 128 or 256. Each row in these arrays, fast time, contains 150 samples spaced 1 μs apart. Each point within a row represents a different range within one pulse. Each column, slow time, contains points sampled the IPP apart, 160 μs for a PRF of 6.25 kHz. Each point within a column represents identical ranges of different pulses. A diagram describing the radar data array is shown in Figure 5.5.

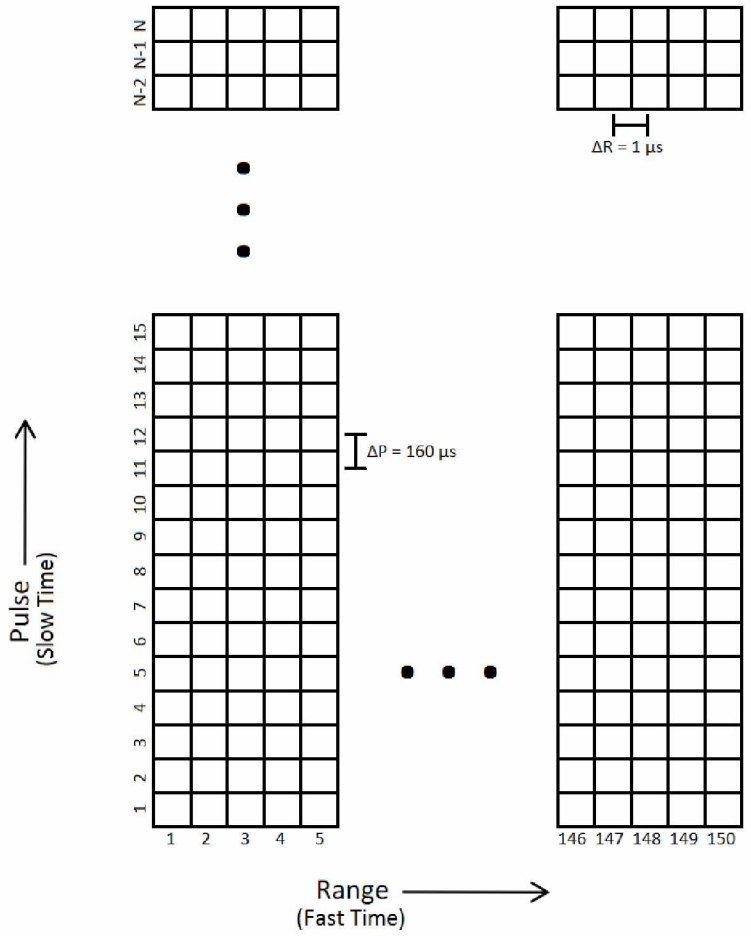

Figure 5.5. Data array for individual azimuths.

5.3.2 Removal of Voltage Discontinuities

The PSTAR system incorporates a feature that was not initially expected, abrupt changes in DC offsets within the received signal. While the purpose of these discontinuities within the PSTAR system is unknown, it is known that they are not compatible with the algorithms presented here. The discontinuities become obvious when plotting a specific range over all pulses, a column (slow time) within the data array is shown in Figure 5.6.

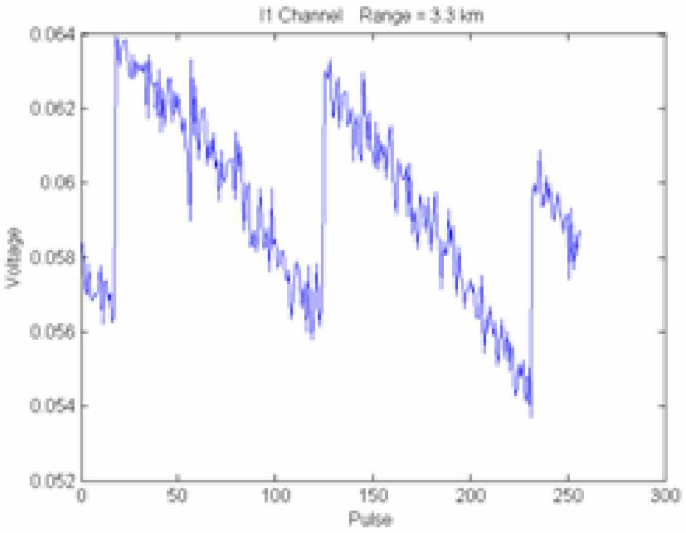

Figure 5.6. DC offset changes.

For reasons that will become clear when discussing the Moving Target Indicator (MTI) in Section 5.4.2.2 these abrupt changes in DC offset must be removed.

To remove the discontinuities they must first be located. The number and position of the discontinuities remains constant within an azimuth, but changes azimuth to azimuth. A plot containing all the data for a channel at a single azimuth is displayed in Figure 5.7.

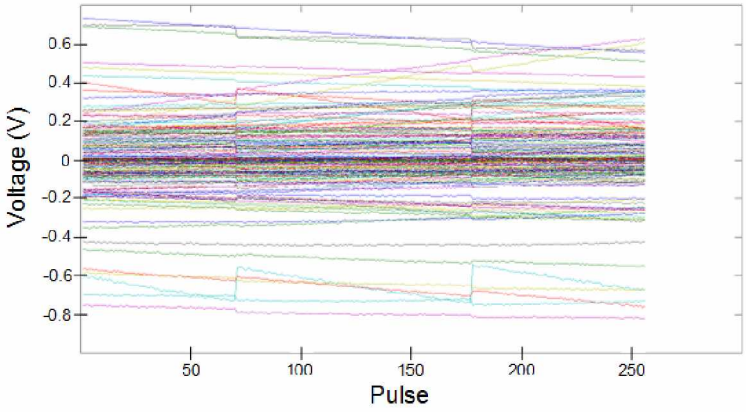

Figure 5.7. Slow time plots of all ranges at a single azimuth before discontinuity removal.

Each trace on this plot represents the data from a single range over all available pulses, 256 in this case. This dataset is used for all examples within Sections 5.3 and 5.4. The abrupt changes are located by taking the absolute value of the numerical derivative of each trace,

$$dI = |I(p) - I(p+1)|. \tag{5.3}$$

All of the ranges are then summed to produce peaks at the locations of any discontinuities present as shown in Figure 5.8.

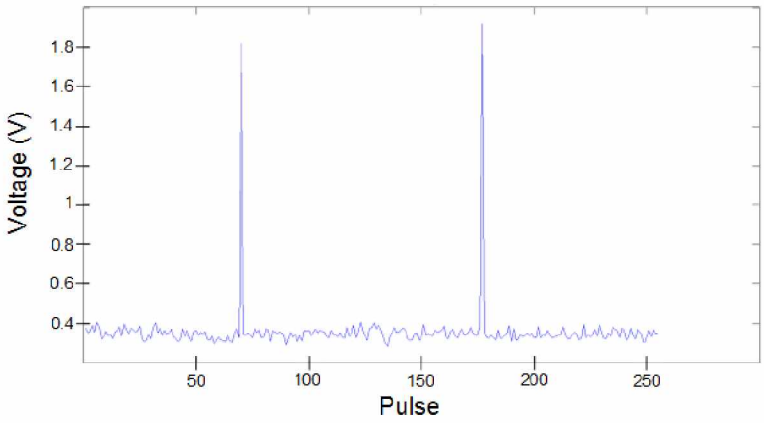

Figure 5.8. Location of DC discontinuities.

The locations of these peaks are determined using the MATLAB *findpeaks()* function with a threshold defined as 1.2 times the mean of the dataset. The value of 1.2 is pulled

from the Discontinuity Threshold constant, **DiscThresh**, defined at the start of the program. With the locations known, the discontinuities are removed using the formula,

$$I(Disc_N + 1 : Disc_{N+1}) = I(Disc_N + 1 : Disc_{N+1}) + [I(Disc_N) - I(Disc_N + 2)] \tag{5.4}$$

with $Disc_N$ representing the location of the Nth discontinuity. Equation 5.4 removes the difference in DC offsets from all data after the first discontinuity. This results in a dataset cleaned of discontinuities as shown in Figure 5.9.

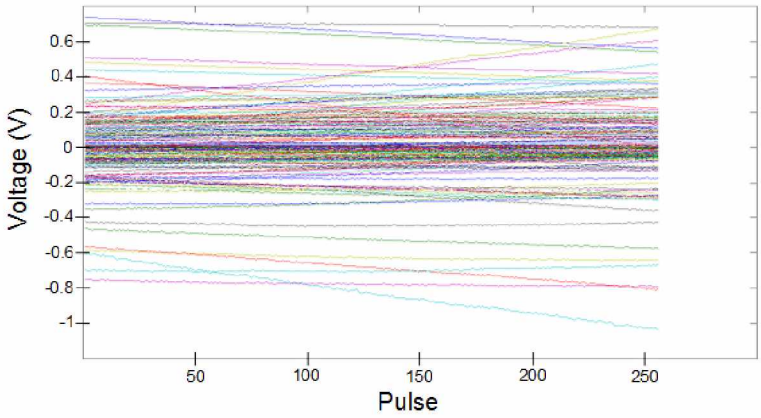

Figure 5.9. Slow time plots of all ranges at a single azimuth after discontinuity removal.

5.4 Target Detection

The transmit signal of a radar can be described by a mathematical function that can be modified to represent the signal at any point in the system. This can be generically represented as a sinusoid modulated by another function, the transmit envelope,

$$S_T(t) = a(t)\cos[\omega_0 t]. \tag{5.5}$$

The pulse envelope used by the PSTAR is a rectangle function bounding a sinusoid of constant frequency,

$$a(t) = A_0 \text{rect}\left(\frac{t - \frac{\tau}{2}}{\tau}\right). \tag{5.6}$$

As such, Equation 5.5 can be re-written as

$$S_T(t) = A_0 \text{rect}\left(\frac{t - \frac{\tau}{2}}{\tau}\right)\cos[\omega_0 t] \tag{5.7}$$

with τ being the pulse width and A_0 being the peak amplitude. Using the transmit power of 1 kW and an impedance of 1 Ω the value of A_0 is 44.72 V. The noise of the transmit signal can be ignored here due to the high SNR.

If the signal reflects off a moving hard point target, such as an airplane, the resulting scattered signal incident on the antenna will be a copy of the transmit signal that has been shifted in time, attenuated, and shifted in frequency as well as containing noise,

$$S_R(t) = a(t - t_R)\alpha_P \cos[(\omega_0 + \omega_d)(t - t_R)] + N(t) \tag{5.8}$$

$$t_R = \frac{2R_T}{c}. \tag{5.9}$$

The constant t_0 is the time delay between the transmission and reception of the signal with R_T being the distance to the target. The variable ω_d is the Doppler frequency of the target causing the reflection. The variable α_P represents attenuation that occurs during propagation and is primarily due to spreading and reflection losses,

$$\alpha_P = \frac{A_R}{A_0} = \sqrt{\frac{P_R}{P_T}}. \tag{5.10}$$

The signal incident on the antenna (Equation 5.8) is amplified by the receiver and mixed to baseband I/Q signals,

$$S_I(t) = a(t - t_R)\alpha_P\alpha_{RX} \cos[(\omega_d)(t - t_R)] + N_I(t) \tag{5.11}$$

$$S_Q(t) = a(t - t_R)\alpha_P\alpha_{RX} \sin[(\omega_d)(t - t_R)] + N_Q(t). \tag{5.12}$$

The variable α_{RX} represents the receiver gain, which is slightly different between the Main and SLC channels,

$$\alpha_{RX} = \frac{A_{RX}}{A_R} = \frac{A_{RX}}{\sqrt{2P_R}}. \tag{5.13}$$

The noise terms, $N_I(t)$ and $N_Q(t)$, account for the noise contained in the sampled signal and are the result of noise sources both internal and external to the PSTAR. The noise terms are assumed to be independent and zero mean.

The infinite time series $S_I(t)$ and $S_Q(t)$ are then digitized with the ADC to produce two discrete time series. The digitization is accomplished by sampling the continuous signals at intervals corresponding to the ADC sampling rate, t_s, and the IPP, T_s,

$$t[m,n] = nt_s + mT_s + \frac{2R_{min}}{c}. \tag{5.14}$$

The independent variables of Equation 5.14, n and m, represent sampling in fast-time and slow-time, respectively. The R_{min} term adjusts for the delay between the start of the transmit pulse and the start of sampling.

5.4.1 Azimuth

Unlike the other target parameters, azimuth information need not be computed from the data. Since the data from each sampled azimuth is separately analysed for the presence of targets, the azimuth of any target detected is indicated by which dataset contains it. The azimuth information contained within each dataset is derived from the time delay between an azimuth zero crossing and when the dataset was sampled. A more thorough explanation can be found in Section 4.1.3.

5.4.2 Range

5.4.2.1 Matched Filter

Each row, fast time, of each of the four signal arrays is now cross-correlated with the transmit pulse envelope. The discrete cross-correlation of two signals, f and g, is defined as

$$(f \star g)[x] \equiv \sum_{y} f^{*}[y]g[x+y]. \tag{5.15}$$

The cross-correlation is implemented with the MATLAB function *xcorr()*. The pulse envelope transmitted by the PSTAR is a rectangle function with a width of 8 μs. For use in the cross-correlation the pulse is represented by an array of 8 ones as it is to be correlated with a signal that was sampled at 1 Msps. For a point target, such as a plane, the reflected signal will be a time delayed copy of the original signal. When cross correlating the transmit pulse with received point target signal, two rectangle functions, the result will be a triangle function with a width twice that of the transmit pulse and a peak at the time corresponding to the range of the target. The uncorrelated plot (Figure 5.10) is of the exact same data array as used to plot Figure 5.9, but with the rows of the array (fast time) were plotted instead of the columns (slow time). This can be compared to the same data after correlation with the transmit pulse shown in Figure **??**. The blue line marks the range at which a target is located.

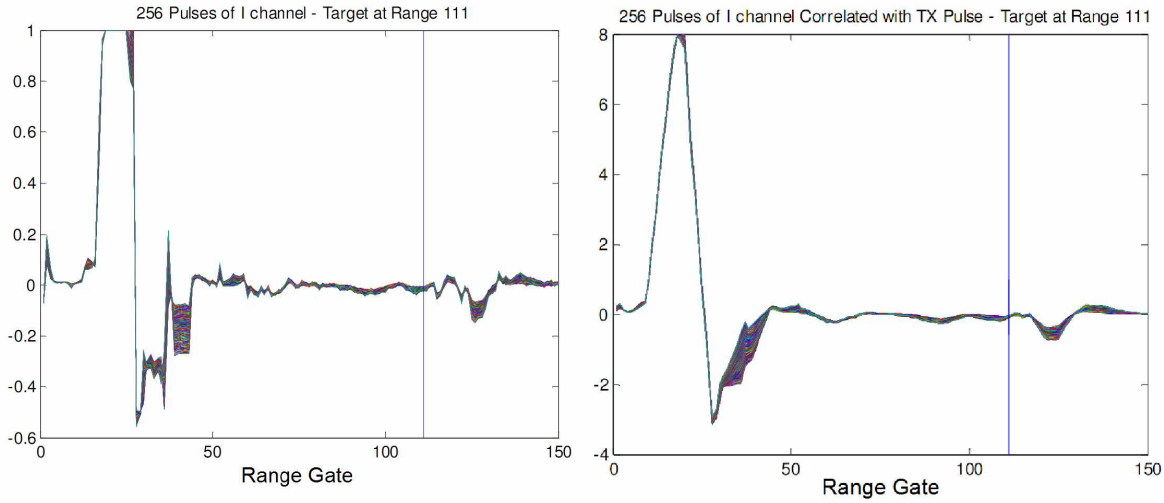

Figure 5.10. Correlation of received fast time signal with transmit pulse. Blue line indicates the presence of a target at that range. Left: Uncorrelated. Right: Correlated.

For the purposes of calculating range the slow-time sampling is unnecessary and m can be defined as constant ($m = 0$). Additionally, the R_{min} term can be ignored now and used to adjust the calculated range later. Making those adjustments to Equation 5.14 yields

$$t(n) = nt_s, \tag{5.16}$$

which can be substituted into Equation 5.11 to produce

$$S_I[n] = a(nt_s - t_R)\alpha_P\alpha_{RX}\cos[(\omega_d)(nt_s - t_R)] + N(nt_s). \tag{5.17}$$

Only the I channel is shown here for brevity; the same procedure applies to the Q channel as well. The sampling rate, t_s, is much greater than the period of the cosine function and so the cosine can be treated as a constant. Lumping it with the attenuation constants we arrive at

$$S_I[n] = a(nt_s - t_R)\alpha + N(nt_s) = \alpha\text{rect}\left(\frac{nt_s - \frac{\tau}{2} - t_R}{\tau}\right) + N(nt_s). \tag{5.18}$$

Substituting $\tau = 8t_s$ produces

$$S_I[n] = \alpha\text{rect}\left(\frac{n - 4 - \frac{t_R}{t_s}}{8}\right) + N(nt_s). \tag{5.19}$$

The $S_Q[n]$ equation will be identical to Equation 5.19 except for the value of the constant α. A matched filter is employed to determine the location of the rectangle function within

the received signal. The matched filter for a given transmit pulse maximizes the signal-to-noise ratio of any copy of the transmit pulse contained within a time-series. A matched filter is a cross-correlation between the conjugate of the transmit pulse envelope and the received data. As the matched filter here is applied to real-valued signals taking the conjugate is irrelevant. Both fast-time signals, $S_I[n]$ and $S_Q[n]$, are cross-correlated with the transmit pulse, $P[n]$, to create a peak at the target range,

$$P[n] = \text{rect}\left(\frac{n-4}{8}\right) \tag{5.20}$$

$$P[n] \star S_I[n] = \left[\text{rect}\left(\frac{n-4}{8}\right)\right] \star \left[\alpha\text{rect}\left(\frac{n-4-\frac{t_R}{t_s}}{8}\right) + N(nt_s)\right] \tag{5.21}$$

$$P[n] \star S_I[n] = \alpha\text{tri}\left(\frac{n-8-\frac{t_R}{t_s}}{16}\right) + N(nt_s). \tag{5.22}$$

Given a positive SNR, the peak of this function, n_R, will be located at the delay associated with the target location,

$$n_R = \frac{t_R}{t_s} = \frac{\left(\frac{2R_T}{c}\right)}{t_s}. \tag{5.23}$$

The target range, R_T, is then found by including the sampling delay offset,

$$R_T = \frac{ct_s n_R}{2} + R_{min}. \tag{5.24}$$

5.4.2.2 Moving Target Indicator

The majority of the power received when operating the PSTAR in a normal setting is from reflections off of stationary ground clutter. Ground clutter can include natural terrain features such as hilltops as well as man-made structures such as buildings. To be able to detect the much lower powered return from a moving aircraft the stationary returns must first be removed from the data. Differentiation between moving targets and ground clutter is performed on the slow time data for each range. An equation describing the slow time data for any range (constant n) can be obtained by sampling Equation 5.11 with

$$t[m] = mT_s. \tag{5.25}$$

Only the I channel is shown here for brevity; the same procedure applies to the Q channel as well. For a range containing just a moving target the result is

$$S_I[m] = \cos[\omega_d m T_s] + N(mT_s). \tag{5.26}$$

For a range containing just a stationary target, such as ground clutter, the result is

$$S_I[m] = \cos[\omega_e m T_s] + N(mT_s). \tag{5.27}$$

For an ideal system ω_e would equal zero and S_I would be a constant. However, the PSTAR is not an ideal system and ground clutter returns produce a signal with a non-zero ω_e. This is likely due to a hardware issue such as a slight difference between the frequency of the transmit pulse and the frequencies used to mix the received signal to baseband. This frequency mismatch causes returns from a stationary target to be a sinusoid just as if the target were moving, however, this sinusoid will have a period much greater than that from an actual moving target. The result is that during the dwell of either 128 or 256 pulses only a fraction of a cycle of the sinusoid produced by a stationary target will be captured. As the percent of the period sampled decreases the more closely that section of the sinusoid can be described by a polynomial function. A section of a sinusoid much less than one period can be reasonably modelled by a 2nd degree polynomial such that at a range containing just stationary ground clutter the received signal can be written as

$$S_I[m] = am^2 + bm + c + N(mT_s) \tag{5.28}$$

with a, b, and c being constants. Ranges containing a moving target often also contain the signal from stationary ground clutter. This case can be described by

$$S_I[m] = (am^2 + bm + c)\cos[\omega_d m T_s] + N(mT_s). \tag{5.29}$$

The presence of slow time ground clutter returns similar in form to 2nd degree polynomials is noticeable in Figure 5.9.

To decrease the presence of the stationary returns a 2nd degree polynomial is individually fit to the slow time data from each range and then the value of the fitted polynomial at each point is subtracted from the original data. For a range containing just ground clutter the subtraction of the fit results in

$$S_I[m] = N(mT_s). \tag{5.30}$$

A range containing both a moving target and stationary ground clutter, after polynomial removal, can be described as

$$S_I[m] = \cos[\omega_d m T_s] + N(mT_s). \tag{5.31}$$

Performing the polynomial removal to all ranges result in a dataset with no DC offsets and contains only moving targets. The data represented in Figure 5.10 is shown after polynomial removal in Figure 5.11.

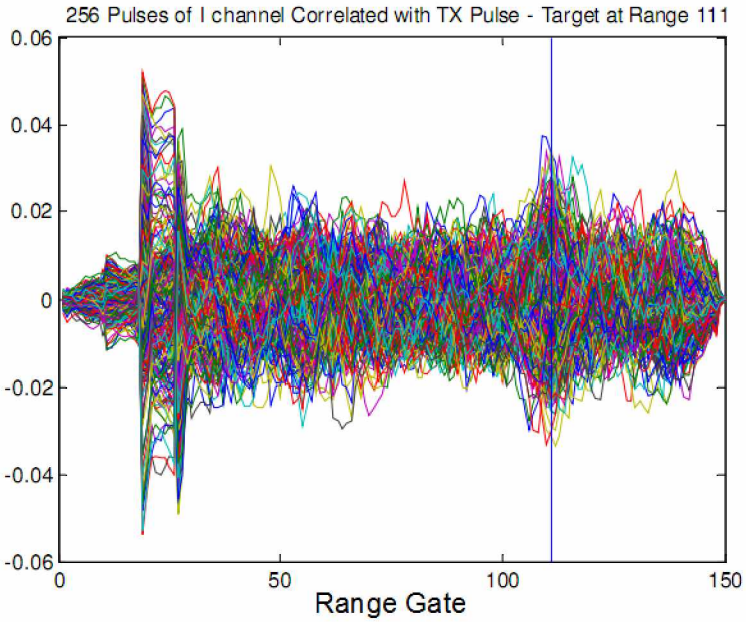

Figure 5.11. Output of MTI.

5.4.2.3 Oversampled Point Target Detection Filter

Up until this point the I and Q signals have been treated as separate real-valued signals, but here they are combined to form a complex signal,

$$S = I + jQ. \tag{5.32}$$

This is done with the I/Q signals from both antennas to form two complex datasets, S_{Main} and S_{SLC}. Any targets present should exist in both datasets, but with a phase difference.

Detection of a target in range only relies on the strength of the signal, not the phase, so the absolute value of the complex signals is taken before the two signals are combined into one through multiplication. This multiplication is done point-by-point using the MATLAB command '.*'; it is not a matrix multiplication. The result is then summed over all the pulses to produce the 1x150 array M (Figure 5.12),

$$M = \sum_{Pulses} |S_{Main}||S_{SLC}|. \tag{5.33}$$

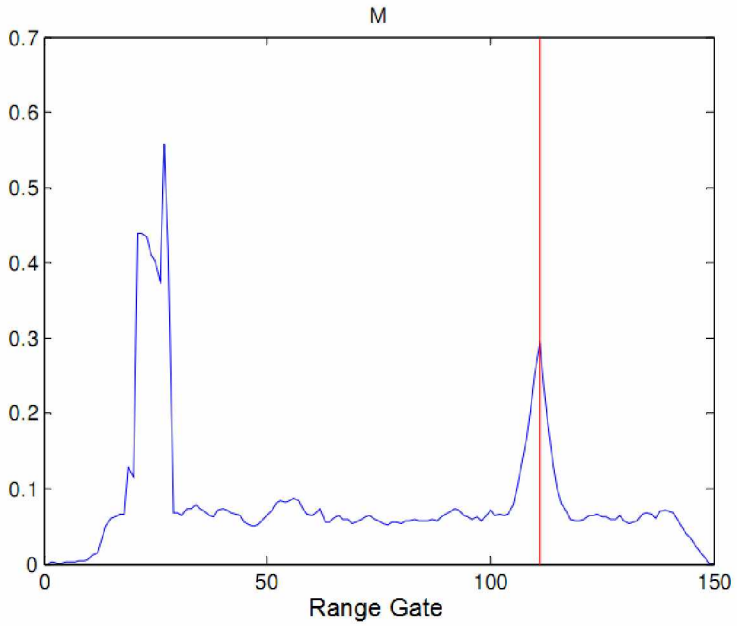

Figure 5.12. Coherently integrated output of MTI.

It is clear from visual inspection of Figure 5.12 that a target is present at the location indicated with a vertical line. If a target is present, every pulse of both *S* arrays will contain a triangle function centered at the target's location, however, the amplitude will vary since the target signal fluctuates pulse-to-pulse as a result of the Doppler shift. Multiplying together a single pulse from both *S* arrays will modify the form of the target signal from a triangle function to something similar to a triangle function, but with the sides no longer linear. As every pulse contains a copy of this non-linear triangle function, summing over all the pulses will increase the amplitude of the non-linear triangle function. All ranges

not containing a target signal will not add coherently and so not be affected as much by the summation.

Since the form of a point target signal is known, a non-linear triangle function, the whole signal can be filtered for the presence of a point target signal. This is accomplished by applying the function

$$M_f(n) = M(n) - W \sum_{k=-L}^{L} \left[M(n+k) - \left(\frac{L-k}{L} \right)^2 M(n) \right] \tag{5.34}$$

to the MTI output M. The variable L is the one-sided length of the filter and the variable W is the weight of the filter. Assuming an ideal target function, the values of $L = 8$ and $W = 1$ would not affect the value of the peak, but all other points would be reduced significantly. In practice, the signal is not ideal and the values of $L = 5$ and $W = 0.9$ were empirically chosen. Using these values, almost all points other than the peak of a target return are reduced to less than zero. This is used to detect the possible presence of a target at all points with a value greater than zero. If two adjacent samples are both greater than zero, then only the higher of the two is selected as a possible target. The data represented in Figure 5.12 is shown after being run through the detection filter (Equation 5.34) in Figure 5.13.

The detection scheme described within this section is based on the assumption that targets will be no less than 1.2 km apart. A detailed analysis of how this detection scheme would be affected by multiple targets separated by less 1.2 km has not been performed. However, it is likely that targets separated by less than 1.2 km may be detected successfully due to the non-maximum length and weight of the filter used, but detectability will decrease as target separation decreases. It may be possible to further facilitate the detection of proximal targets by altering the weight and length of the filter or by modifying the filter shape.

This method of target detection occasionally produces false-positives, but these are screened out in the Doppler processing. The ranges in the extremes, both near and far, are the most likely to produce false positives due to the inability to apply Equation 5.34 over the full range of the summation. The exact probabilities of detection for this system are unknown and would required more comprehensive testing to determine. Analytical calculation of probabilities of detection was not performed due to the complexities arising from a few issues. The first is that the dominant source of unwanted power in the signal

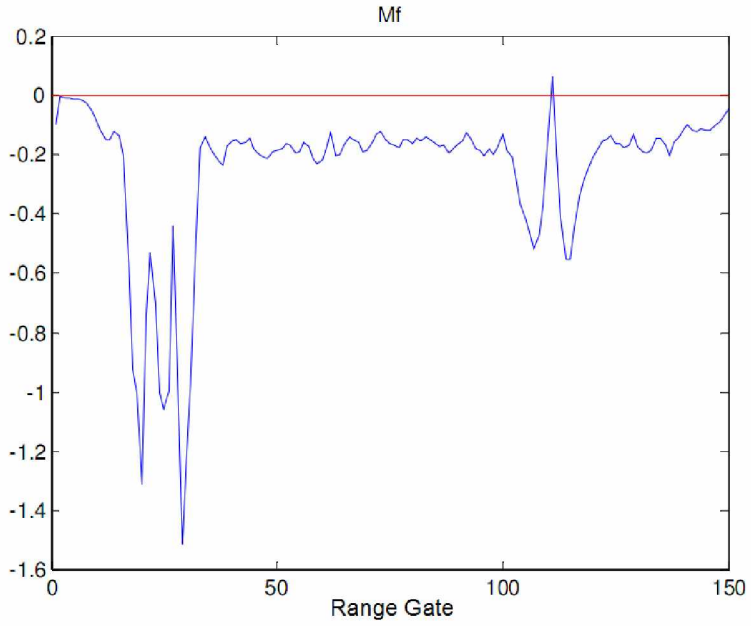

Figure 5.13. Output of range detection filter.

is not noise, but remnants of ground clutter that were not removed by the MTI. Clutter signals are non-uniform and will vary depending on the terrian in which the system is operated. Clutter at close ranges can occasionally cause the receiver. become saturated. Any pulse containing saturated ranges will contribute nothing to the combined pulse SNR of a target at that range. Another issue impeding the analytic calculation of detection probabilities is that the removal of discontinuities (Section 5.3.2) is not perfectly clean and occasionally leaves a spike larger than the surrounding noise level. A final difficulty is the fact that since the signal is oversampled the target signal from multiple adjacent range cells is used in the detection resulting in a different SNR than when analysing range cells individually.

5.4.3 Doppler Velocity

The slow time data for all ranges containing a possible target is now analysed for the presence of a target in the Doppler spectrum. The Doppler spectrum is computed by taking

the discrete Fourier transform of the complex slow time data from S_{Main},

$$D(k) \equiv \sum_{y=0}^{Y-1} S(y) e^{-\frac{j2\pi}{Y} ky}. \tag{5.35}$$

The signals S_{Main} and S_{SLC} cannot be combined here due to the phase difference present in the two. The discrete Fourier transform is performed using the MATLAB function *fft()*. If a moving target is present at the selected range the slow time data will contain a function of the form $e^{j2\pi ft}$ with a frequency equal to the Doppler frequency of the target. Taking the Fourier transform of this we get

$$\mathcal{F}\{e^{j2\pi f_0 t}\} \rightarrow \delta(f - f_d). \tag{5.36}$$

The location of the resulting delta function within the Doppler spectrum indicates the frequency of the original signal. If there is no moving target present at the selected range there will be no sharp peak in the Doppler spectrum due to the absence of the delta function produced by the target signal.

Computing the Doppler frequency of a target depends on the slow-time samples of a signal. The slow-time sampling is performed by sampling Equations 5.11 and 5.12 with

$$t[m] = mT_s, \tag{5.37}$$

which produces

$$S_I[m] = \cos[\omega_d mT_s] + N(mT_s) \tag{5.38}$$

$$S_Q[m] = \sin[\omega_d mT_s] + N(mT_s). \tag{5.39}$$

These two real functions can be combined into a single complex function,

$$S[m] = S_I[m] + jS_Q[m] = e^{j2\pi f_d mT_s} + N(mT_s) + jN(mT_s). \tag{5.40}$$

The discrete Fourier transform is than applied to Equation 5.40,

$$D[f] = \mathcal{F}\{S[m]\} = \delta[f - f_d] + N(mT_s). \tag{5.41}$$

The peak of this function, f_d, corresponds to target Doppler frequency.

There may still be the residual traces of stationary ground clutter in the Doppler spectrum, so all points less then the minimum Doppler velocity are set to the mean of the

Doppler spectrum. The presence of a delta function is then determined using a threshold. The threshold is defined by the **DoppThresh** constant, set at 10, times the mean of the Doppler spectrum. If a peak is present the range is considered to have a legitimate target with a Doppler velocity indicated by the location of the peak. The target shown in Figure 5.14 has a Doppler velocity of -43.7 m/s.

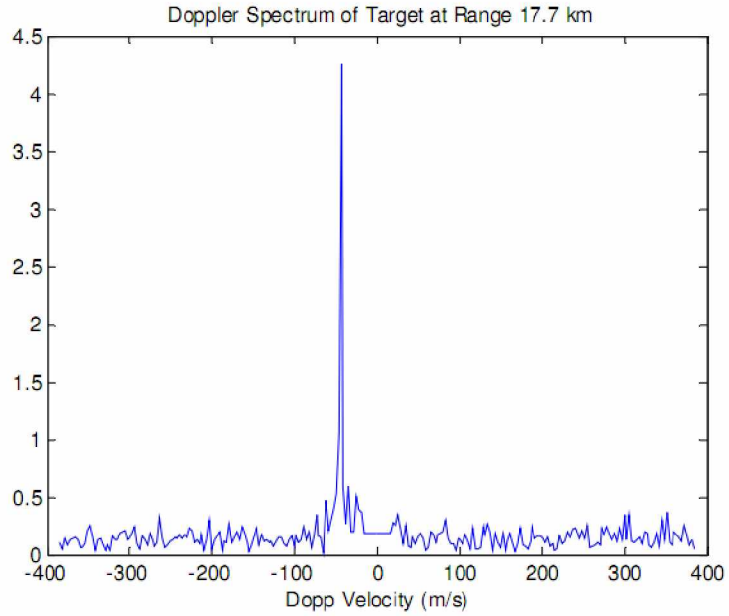

Figure 5.14. Doppler spectrum of target.

The detection scheme described within this section assumes each range contains only a single target. Any ranges containing multiple targets will have, at best, only one detected. In addition to multiple discrete targets, helicopters will produce multiple peaks within the Doppler spectrum as a result of blade flash, reflections off the rotating blade. Any helicopter targets detected will produce erroneous velocities due to the measured Doppler velocity being a combination of the helicopter's total velocity and the blade's rotational velocity.

5.4.4 Elevation Angle

The last step of the target detection is to determine elevation angle. This is the most processor intensive portion of the detection algorithm, due to curve fitting, but it is only done when a target has been detected successfully in both range and Doppler, minimizing the amount of curve fitting to be done. The process of determining elevation angle is similar to that used in the calibration. A sine wave is fit to each of the four I/Q signals independently (Equation 5.1, Figure 5.15).

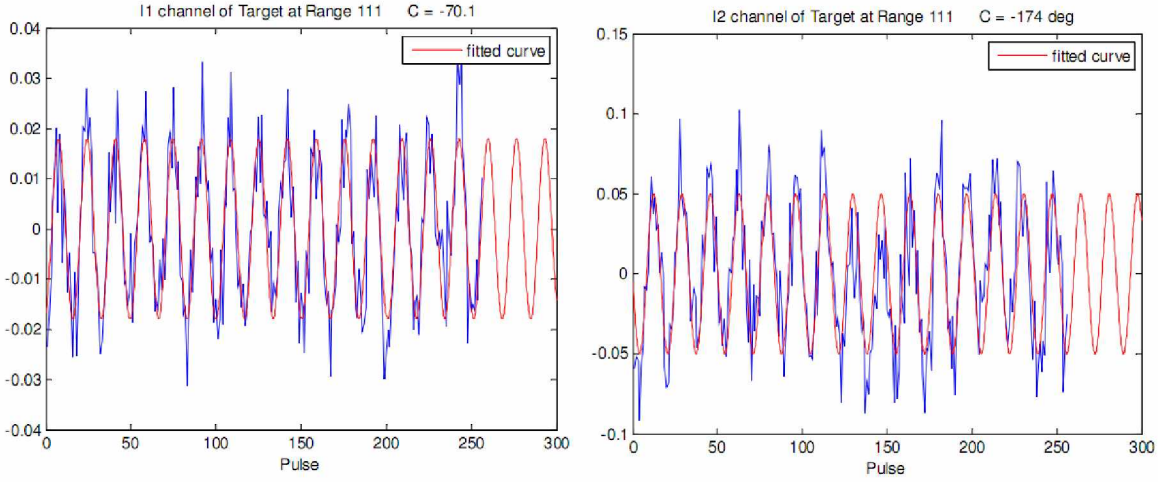

Figure 5.15. Curve fitting of slow time target signals. Left: Main channel. Right: SLC channel.

The complex signals are not used because they inherently assume that the I and Q signals have identical amplitudes and an ideal 90° phase difference. This is not the case due to the non-ideal nature of physical hardware and treating them as separate real-valued signals removes the need for those assumptions. The only parameter of the fit needed is the phase offset, ϕ. The difference in phase between the Main and SLC channels is computed using both the I and Q signals,

$$\Delta\phi_I = \phi_{I1} - \phi_{I2} \tag{5.42}$$

$$\Delta\phi_Q = \phi_{Q1} - \phi_{Q2}. \tag{5.43}$$

The target phase difference is determined by subtracting the channel phase difference, $\Delta\phi_0$,

which was determined in the calibration, from the average of I and Q phase differences,

$$\Delta\phi_T = \frac{\Delta\phi_I + \Delta\phi_Q}{2} - \Delta\phi_0. \tag{5.44}$$

Possible elevation angles can then be determined using the target phase difference,

$$\text{Elevation Angles} = \sin^{-1}\left[\left(k + \frac{\Delta\phi_T}{2\pi}\right)\left(\frac{\lambda}{d}\right)\right]. \tag{5.45}$$

The variable λ is the wavelength, 0.246 m, and d is the antenna spacing, 0.48 m. Due to ambiguity, there are multiple possible elevation angles that could produce the measured $\Delta\phi_T$. The most probable elevation angle is chosen as the minimum positive elevation angle. This is a valid assumption due to the 3-dB vertical beam width of the antenna being about 3° less than the first ambiguous elevation angle.

To determine target elevation angle from a dataset, the signals sampled in slow-time by both antennas must be compared,

$$S_I^{Main}[m] = \alpha\cos[\omega_d m T_s] + N(mT_s) \tag{5.46}$$

$$S_I^{SLC}[m] = \alpha\cos[\omega_d m T_s - \phi_0] + N(mT_s). \tag{5.47}$$

Where ϕ_0 is the phase difference caused by the elevation angle, θ_T, of the target,

$$\phi_0 = \frac{2\pi d}{\lambda}\sin(\theta_T). \tag{5.48}$$

A curve of form $\cos[\omega_0 m T_s - \phi]$ is fit to the data from both channels to determine the phase offsets ϕ^{Main} and ϕ^{SLC}. The target phase difference, ϕ_T, is found by computing the difference of the two phase offsets,

$$\phi_T = \phi^{Main} - \phi^{SLC} = \frac{2\pi d}{\lambda}\sin(\theta_T). \tag{5.49}$$

The target elevation angle is then computed,

$$\theta_T = \sin^{-1}\left(\frac{\phi_T\lambda}{2\pi d}\right). \tag{5.50}$$

The method of determining target elevation angle described within this section assumes each range contains only a single target. Ranges containing multiple targets will contain multiple sinusoids and so the single sinusoid fit used here would not adequately

describe the signal. It may be possible to extend the fit to account for the presence of multiple targets, but that would substantially increase processing time. Using the Doppler velocity as a means of narrowing down the set of possible fits could be used to help mitigate the increase in complexity arising for the presence of multiple sinusoids within the data.

During development, an alternate method of determining target elevation angle was explored. This was to cross-correlate the Main and SLC channel data to determine the time offset between them. Using the already computed Doppler frequency, the time offset can be used to estimate the phase offset. This approach was computationally faster than the curve fitting approach presented in this section, but it was not as accurate. Some of this decreased accuracy comes from the use of Doppler velocity which itself is not exact. An additional issue was that performing a cross-correlation between the two channels assumes that the noise contained in both channels is independent, which may not be a valid assumption.

5.5 Processing Demand

Due to the radar data being post-processed there were not strict processing time constraints placed on the algorithm as there would be for a real-time system. Despite this, it is still interesting to investigate which portions of the algorithm consume the largest amounts of processing time. This was determined using the MATLAB Profiler which tracks the amount of time spent on functions within a MATLAB program. The test used a 5° azimuth spaced dataset containing only one target and the dialog boxes normally presented to the user were suppressed to eliminate any delays due to user interaction. A summary of the results can be found in Table 5.3.

The total time required to process a data file, which contains a single sweep, was a little under a minute while the actual time span represented by the dataset is about 6 s. This results in a processing time requirement of about tenfold more than the time duration of data being processed. The most processor intensive function was RemoveFit which is effectively the MTI. Within this function most of the time, 81%, was spent executing the MATLAB function *polyfit()*. One way in which the execution speed could be increased is by decreasing the complexity of the function being fit to the data. The second degree

Table 5.3. Summary of execution time for a data file containing a single target.

Function	Calls	Total Time (s)	Time/Call (s)	% Time
RemoveFit.m	144	27.24	0.19	51.1
Correlate.m	72	16.08	0.22	30.2
Calibrate.m	1	2.21	2.21	4.1
LoadTDMS.m	1	1.96	1.96	3.7
Fix_Discontinuity.m	288	1.91	0.01	3.6
Find_Discontinuity.m	288	1.34	0.01	2.5
PhaseDifference.m	5	1.31	0.26	2.5
All Others	-	1.23	-	2.3
Total	-	53.28	-	100

polynomial currently being used could be replaced with a first degree polynomial, but that is as simple of a function that could be used before eliminating the usefulness of the curve fitting based MTI. To drastically decrease the time spent on the MTI it would be necessary to eliminate the need for curve fitting. The MTI is essentially just a high-pass filter, and one could be implemented using standard DSP techniques. This is commonly implemented in radar systems using a N-pulse canceller, and this approach was investigated early in the development of the algorithm presented here. It was abandoned due to not being as powerful as the curve fitting based MTI.

The Correlate function consumes about 30% of the total execution time, but it is doubtful that this could be decreased much without drastically changing the system. Most of the time, 89%, consumed by Correlate is used by the *xcorr()* function. This function has been optimized by the designers of MATLAB and is more efficient than the author of this thesis could implement a cross-correlation function in MATLAB. The only apparent solution is to perform the correlation in real-time before the data is stored, but that is beyond the scope of this thesis. The *Calibrate.m* function used 4.1% of the total time, but this was only for one sweep. *Calibrate.m* only needs to be run once for a batch of data files so the fraction of total processing time used would decrease significantly if more than one data file is processed at a time. *Fix_Discontinuity.m* and *Find_Discontinuity.m* could be optimized more if

desired, but they correct for a feature of the PSTAR for which it may be possible to remove in hardware. If the PSTAR were to be modified to not produce DC discontinuities then the *Fix_Discontinuity.m* and *Find_Discontinuity.m* functions could be eliminated entirely. The *PhaseDifference.m* function is very processor intensive due to the four sinusoid fits that are computed during each call as can be seen by the high time per call. The only functions with higher times per call are *Calibrate.m* and *LoadTDMS.m* which are only called once per batch and once per rotation, respectively. The time consumed by *PhaseDifference.m* could be decreased by only executing it for confirmed targets, but the current implementation executes it for all possible targets. In the dataset represented in Table 5.3 this would result in a 80% decrease in processing time because there was only one confirmed target in the dataset out of the five possible targets detected.

Chapter 6
Testing and Verification

6.1 Simulated Data

A useful method for testing the data processing algorithm is to supply it with simulated data. The MATLAB program used to create the simulated data, *Sim_Data.m*, can be found in the Section A.2.1. This program creates a dataset containing I/Q voltage data for both the Main and SLC channels for a single dwell. Targets with arbitrary range, Doppler velocity, elevation angle, and RCS can be included as well as an arbitrary amount of phase jitter and Gaussian random additive noise power. There is also the option to add receiver DC, amplitude, and phase offsets.

Simulated datasets were created containing target signals of known SNR. These datasets were used to determine the minimum SNR required for detection. The threshold for detection occurs at an SNR of about -21 dB per pulse when using a dwell of 128 pulses, which corresponds to an SNR of about 0.1 dB after summing over all pulses. The MTI output of data containing a target signal with an SNR at this cutoff can be seen in Figure 6.1. This can be compared to the MTI output of a real dataset shown in Figure 5.11. The detection threshold could be further decreased in the simulated data because the target signal present in the MTI output (Equation 5.33) is still well above the noise floor as seen in Figure 6.2. Real data differs from the simulated data in that it contains more than just a target signal in the presence of zero mean Gaussian noise. The noise present in actual data is likely to be neither zero mean nor Gaussian and there will also be the presence of ground clutter returns and possibly interference. Various algorithmic constants were empirically chosen to tune the detection of a target within real data and while they could be re-tuned for the detection of low-SNR targets within simulated data this would increase the number of false detections in real data.

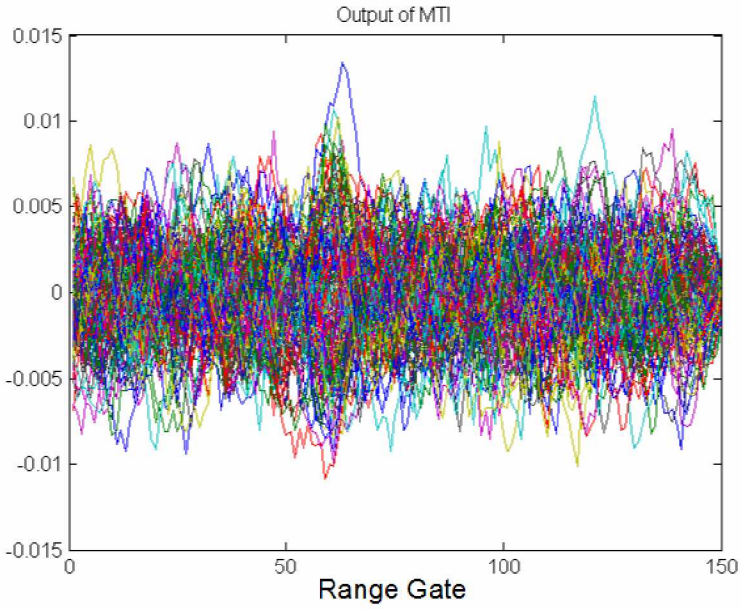

Figure 6.1. Output of MTI for a simulated target with a per pulse SNR of -21 dB.

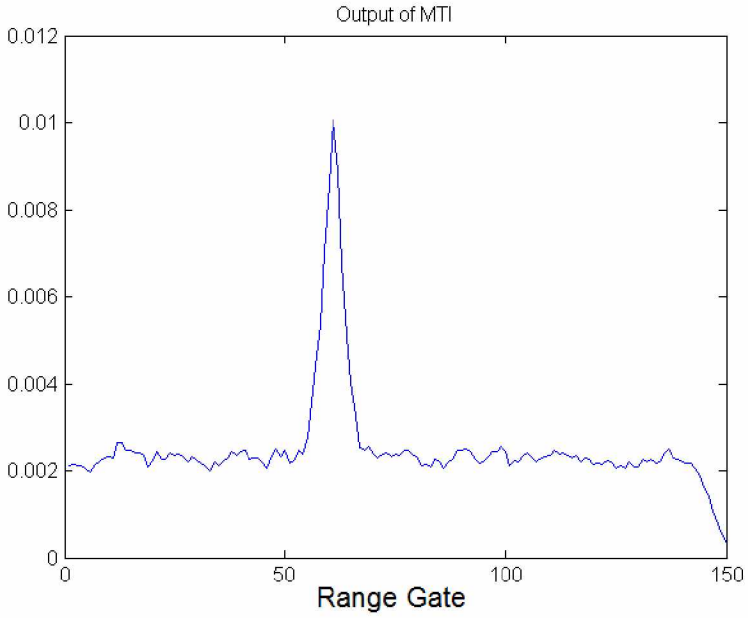

Figure 6.2. Coherently integrated output of MTI using 128 pulses of simulated data with a single pulse target SNR of -21 dB.

Another use for the simulated data is to assist in developing a method to relate a feature of a target signal to the received power of the signal. For real targets the received signal power is unknown, but can be estimated using Equation 3.1. This estimation is complicated by the fact that the RCS of the target is unknown and variable in time, due to temporal changes in orientation. The received signal power of a target is well defined in simulated data and so a formula relating the received signal power to some feature of the target signal can be determined. The target signal feature chosen was

$$X = \max\left(\sum_{pulses} |\Re\mathfrak{e}\{\mathbf{S1f}\}|\right). \tag{6.1}$$

The array **S1f** contains the correlated complex signal from the main channel and is equivalent to the variable S in Equation 5.32. The corresponding value of X was determined for a variety of values of target signal power, P_R, as listed in Table 6.1.

Table 6.1. Values of X for various received target powers.

P_R (W)	X
10^{-6}	40.72
10^{-7}	12.88
10^{-8}	4.072
10^{-9}	1.288
10^{-10}	0.4072
10^{-11}	0.1288
10^{-12}	0.0407
10^{-13}	0.0129

These values were used to empirically determine a formula that relates a known value of X to the corresponding value of P_R,

$$P_R = \frac{X^2}{16.58 \text{ x } 10^8}. \tag{6.2}$$

Equation 6.2 can be used to determine the received signal power of a target within real data. If the propagation losses were accurately modelled the knowledge of received signal power could be used to estimate the physical size of a target based on its RCS.

6.2 Cooperative Target Tests

Testing of the radar system can be aided by use of cooperative targets. The cooperative target used for these tests was a small single engine plane flying radial patterns over the radar while carrying a GPS logger.

Two datasets from a cooperative target flight were obtained. One was sampled at 10° azimuth increments with 256 pulses per dwell, and the other with 5° azimuth increments with 128 pulses per dwell. These datasets were useful in further refining threshold values used in the data processing algorithm. Figure 6.3 contains the target hits from the 10° azimuth data over a time period of about 15 minutes. The color of the target X's indicates the time of the target hit, blue indicates the beginning of the dataset and red indicates the end. Each range ring is 5 km.

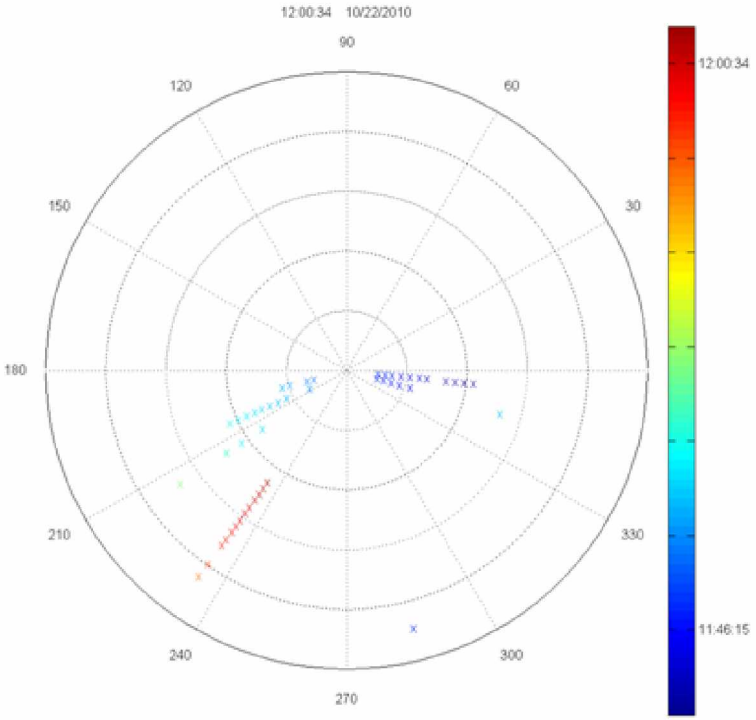

Figure 6.3. Target hits from a cooperative target with 10° azimuth spacing.

An issue that is apparent here is that the radar detects the same target in adjacent azimuths. In the 10° data there is very rarely more than two adjacent azimuths that detect the

same target so a simplistic filtering method of ignoring the weaker of the two returns can be implemented with fairly successful results. The 5° data presents a more complicated problem as there are often three, and occasionally four, adjacent azimuths that contain the same target. The presence of duplicate target hits at adjacent ranges is made clear in the 5° data over about 45 minutes seen in Figure 6.4.

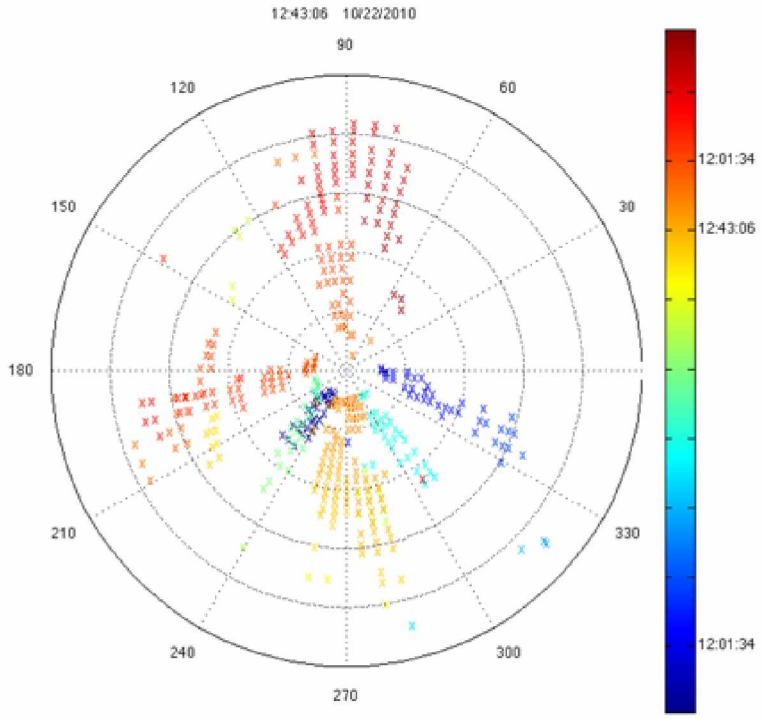

Figure 6.4. Target hits from a cooperative target with 5° azimuth spacing.

An intelligent method for determining the target hit that most accurately represents the target's position would be useful, perhaps using a Kalman filter. However, that is beyond the scope of this thesis. It was decided to include all valid target hits in the dataset as the velocity and elevation angle data of off bore-sight target hits may be useful for improving the accuracy of the bore-sight target hit.

When comparing the GPS logger data to the radar data it became apparent that both the north alignment and time synchronization were faulty. It was determined that the azimuth produced by the radar was about 110° off and the time by 3:01:53. The azimuth error may

be due to a north alignment not being performed immediately prior to data collection as it was assumed that the one done previously was still valid as the PSTAR pedestal had not been moved. The time difference was a surprise as the system time was synchronized to within a second of GPS time. However, the timestamp produced by the ADC card does not match the system time resulting in the time error. The radar data were rotated in azimuth and shifted in time to align to with the GPS data for a comparison of the two. To do this comparison it was needed to remove hits from the same target at adjacent azimuths. As this task has not been automated, it was done manually by selecting the target hit which is closest in azimuth to the actual plane as determined by the GPS data. The use of azimuth for this is a little troublesome due to the ad hoc north alignment, but it is good enough for a rough comparison, which can be seen in Table 6.2. The 5° spaced data contains 76 unique target hits and the 10° spaced data contains 37 unique target hits.

Table 6.2. Comparison between GPS and radar data.

5° Azimuth Spacing

	Range Error (m)	Az Error (°)	Elev Angle Error (°)	Vel Error (m/s)
Max	649	5	15	105
Min	-163	-6	-22	-31
Mean	282.3	-0.4	-2.6	-0.9
St. Dev.	284.6	2.3	5.9	13.5

10° Azimuth Spacing

	Range Error (m)	Az Error (°)	Elev Angle Error (°)	Vel Error (m/s)
Max	619	5	13	8
Min	-183	-8	-13	-29
Mean	159.1	-0.2	0.8	-2.9
St. Dev.	278.2	2.7	5.9	6

The accuracies of most target parameter estimations are constrained by quantization errors arising from sampling rates. The azimuth is sampled at both 5° and 10° increments resulting in expected errors of ±2.5° and ±5°, respectively. The range error expected from the system is due to the fast time sampling rate of 1 μs, resulting in a separation of 150 m

between range gates (See Equation 3.5). This spatial sampling period of 150 m provides an expected range accuracy of ± 75 m. The spacing of Doppler bins is governed by both the PRF and the number of pulses per dwell (See Equation 3.8). The velocity accuracy of this system is expected to be ± 3 m/s and ± 1.5 m/s for data sampled with 128 pulses per dwell and 256 pulses per dwell, respectively. The accuracy of elevation angle estimations is not limited by quantization error in the same way that the other target parameters are. The elevation angle estimation relies on fitting a continuous time function to the discrete data and the number of elevation angle bins is limited only by the quantization error inherent in the variable type of the data, which is sufficiently small that it can be ignored. A comparison of GPS and radar data is shown in Figures 6.5 through 6.9. The red line on each of these plots indicates perfect agreement between GPS and radar data and the blue lines indicate error bounds. For the azimuth and velocity plots (Figures 6.5 and 6.7) the error bounds are equal to the quantization error. The error bounds on the range plot (Figure 6.6) are set at ± 500 m due to the quantization error of ± 75 m being too small to be noticeable on the plot. The error bounds included in the elevation angle plots (Figures 6.8 and 6.9) are arbitrarily set at $\pm 2.5^{\circ}$.

The measured azimuth is in good agreement with GPS data as shown in Figure 6.5 with almost all radar data lying within the expected error bounds. The measured range shows a strong correlation between radar and GPS measurements, however, there are a large number of target hits with an error of about 500 m which is well beyond the expected error of 75 m (Figure 6.6). A comparison between Radar and GPS calculated Doppler velocity can be seen in Figure 6.7. It can bee seen that the majority of inward travelling target hits (positive Doppler velocity) lie within the error bounds. Outward travelling targets show a tendency to have an estimated Doppler velocity slightly less then the true radial velocity. With sufficient testing it may be possible to compensate for this effect if it proves to be consistent.

The calculated elevation angle has the least accuracy of all the target parameters estimated by radar. Plots comparing the radar measured elevation angle to that computed from GPS coordinates can be seen in Figure 6.8. It can be seen that while there is a positive correlation between the radar and GPS calculated elevation angles the accuracy of the determination of elevation angle from radar data would need to be improved before this

system could be utilized for the intended purposes of the FAA and PFRR.

In the case of the 5° spaced data there are often multiple adjacent azimuth bins that contain reflected power from the same target. A method of improving the elevation angle calculation is to use data from some, or all, of the duplicate targets that were ignored in this analysis. For targets with multiple detections within a single sweep, the detection with an estimated elevation angle closest to that computed from GPS data was selected and included in Figure 6.9. This shows an increase in elevation angle accuracy could be possible if a method were devised to incorporate the multiple detections of targets oversampled in azimuth.

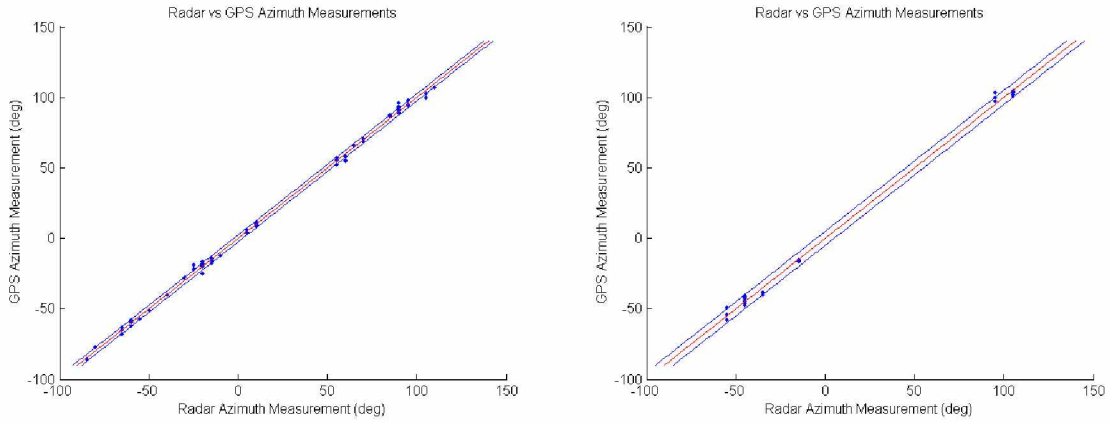

Figure 6.5. Comparison of GPS and radar azimuth measurements. Left: Azimuth spacing of 5°. Right: Azimuth spacing of 10°.

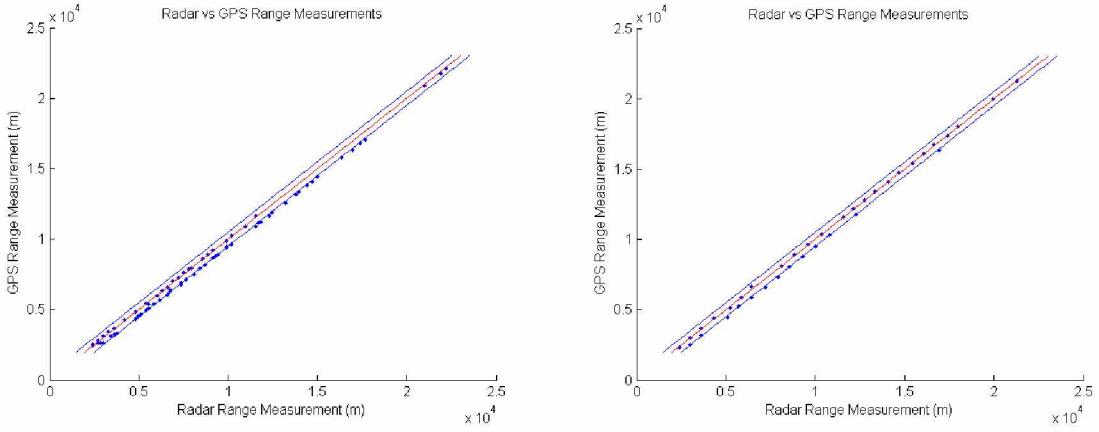

Figure 6.6. Comparison of GPS and radar range measurements. Left: Azimuth spacing of 5°. Right: Azimuth spacing of 10°.

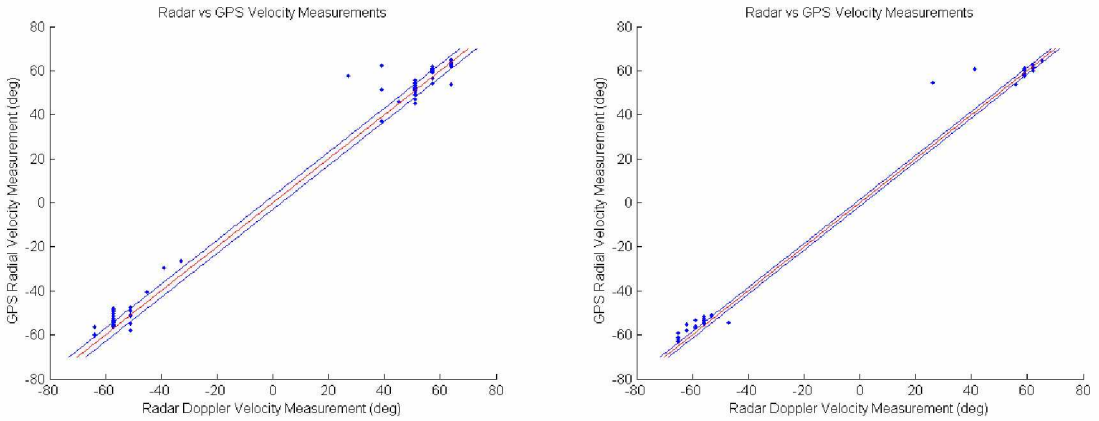

Figure 6.7. Comparison of GPS and radar velocity measurements. Left: Azimuth spacing of 5°. Right: Azimuth spacing of 10°.

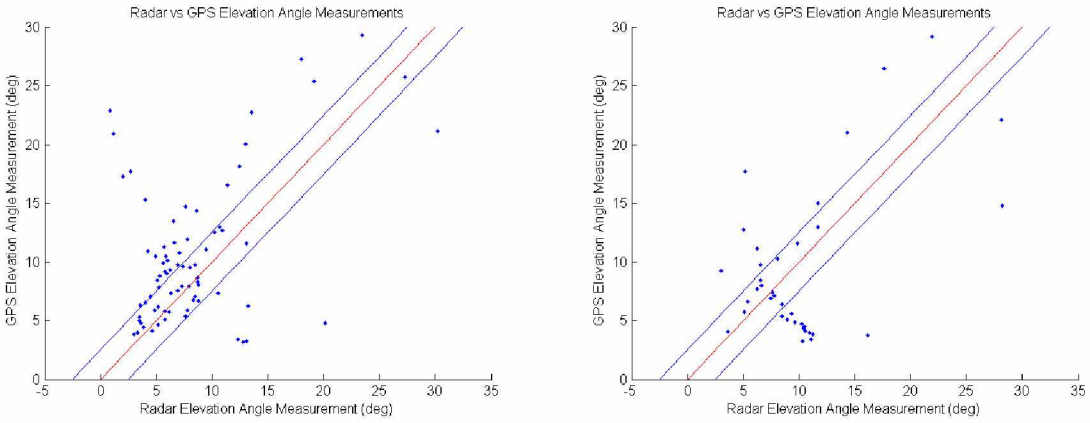

Figure 6.8. Comparison of GPS and radar elevation angle measurements. Left: Azimuth spacing of 5°. Right: Azimuth spacing of 10°.

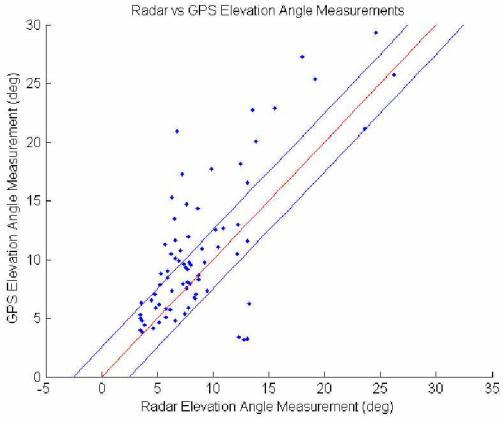

Figure 6.9. Comparison of GPS and radar elevation angle measurements for an azimuth spacing of 5° using target data with the closest elevation angle.

Chapter 7
Conclusions and Future Improvements

The system that has been described in the preceding chapters is more of a proof of concept than a functional system to be deployed by the FAA or PFRR. In this chapter we will examine future improvements that could be made to increase the utility of the system.

7.1 Unused Algorithms

Throughout the process of developing this system there where avenues pursued that, for one reason or another, did not make it into the final design. Some avenues, such as triangulation to determine elevation angle, were considered, but ruled out as a viable option before implementing them. Others, such as the original two methods of procuring azimuth data, were developed to at least some degree of implementation, but eventually abandoned in favor of a more useful approach. There were two features that were investigated and implemented fairly thoroughly, but eventually abandoned, that may be of use in future development of this system. These features are a more detailed calibration than the one presented in Section 5.2 and a cluttermap. These features were originally developed because the author was essentially grasping at straws in a vain attempt to detect targets in the data before the presence of DC discontinuities (see Section 5.3.2) was discovered. The presence of the discontinuities interfered with the proper functioning of the MTI, resulting in ground clutter dominating the signal and drowning out any airplane echos. Development of a cluttermap was pursued as a means of mitigating the ground clutter that was inexplicably not being removed properly by the MTI. Development of a detailed calibration routine was done concurrently with development of the cluttermap, and both were abandoned when removal of the discontinuities allowed for the proper functioning of the MTI.

7.1.1 Detailed Calibration

The calibration procedure described in Section 5.2 provides the necessary channel phase difference information, but the original calibration scheme developed provides a much more complete characterization of the receiver. As discussed in Section 2.2, many of the PSTAR receivers tested produced notable errors in the I/Q outputs and only the most

functional were selected for use. However, the receivers chosen for use in development still contained small errors including DC offsets, non-ideal I-Q phase offsets, and amplitude imbalances. A detailed calibration routine was developed to mitigate these errors, small as they may be in the good receivers, when no other path towards successful target detection was evident to the author. While the detailed calibration is not necessary when operating a PSTAR with a good receiver, it may be useful in utilizing receivers that were previously discarded due to large errors. The following is an implementation of a procedure described by Richards [20].

The detailed calibration starts with the I and Q signals described in Equations 5.11 and 5.12. Here we will assume noiseless signals, but will include receiver non-idealities. These include an amplitude imbalance (ε), a phase difference (ϕ_R), and DC offsets (O_I and O_Q),

$$S_I(t) = a(t-t_0)\alpha_P\alpha_{RX}\cos[(\omega_d+\omega_e)(t-t_0)] + O_I \tag{7.1}$$

$$S_Q(t) = \varepsilon a(t-t_0)\alpha_P\alpha_{RX}\sin[(\omega_d+\omega_e)(t-t_0)+\phi_R] + O_Q. \tag{7.2}$$

We can clean up Equations 7.1 and 7.2 by defining

$$B = a(t-t_0)\alpha_P\alpha_{RX} \tag{7.3}$$

$$\theta = (\omega_d+\omega_e)(t-t_0). \tag{7.4}$$

Substituting these back into Equations 7.1 and 7.2 we get

$$S_I(t) = B\cos\theta + O_I \tag{7.5}$$

$$S_Q(t) = \varepsilon B\sin(\theta-\phi_R) + O_Q. \tag{7.6}$$

We then define the signals S_I' and S_Q' as ideal I/Q signals with no offsets or imbalances,

$$\begin{bmatrix} S_I' \\ S_Q' \end{bmatrix} = \begin{bmatrix} B\cos\theta \\ B\sin\theta \end{bmatrix}. \tag{7.7}$$

The ideal I/Q signals can be related to the sampled I/Q signals through a subtraction of the DC offsets (O_I and O_Q) and a matrix multiplication with **C**,

$$\begin{bmatrix} S_I' \\ S_Q' \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \left(\begin{bmatrix} S_I \\ S_Q \end{bmatrix} - \begin{bmatrix} O_I \\ O_Q \end{bmatrix} \right) \tag{7.8}$$

$$\begin{bmatrix} B\cos\theta \\ B\sin\theta \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} B\cos\theta \\ \varepsilon B\sin(\theta-\phi_R) \end{bmatrix}. \tag{7.9}$$

The elements of the **C** matrix are obtained by solving Equation 7.9 for $B\cos\theta$ and $B\sin\theta$,

$$B\cos\theta = c_{11}B\cos\theta + c_{12}\varepsilon B\sin(\theta-\phi_R) \tag{7.10}$$

$$c_{11}B = B \tag{7.11}$$

$$c_{11} = 1 \tag{7.12}$$

$$c_{12}\varepsilon B = 0 \tag{7.13}$$

$$\varepsilon, B \neq 0 \tag{7.14}$$

$$c_{12} = 0 \tag{7.15}$$

$$B\sin\theta = c_{21}B\cos\theta + c_{22}\varepsilon B\sin(\theta-\phi_R) \tag{7.16}$$

$$B\sin\theta = c_{21}B\cos\theta + c_{22}\varepsilon B[\sin\theta\cos\phi_R - \cos\theta\sin\phi_R] \tag{7.17}$$

$$B\sin\theta = (\cos\theta)(c_{21} - c_{22}\varepsilon B\sin\phi_R) + (\sin\theta)(c_{22}\varepsilon B\cos\phi_R) \tag{7.18}$$

$$B = c_{22}\varepsilon B\cos\phi_R \tag{7.19}$$

$$c_{22} = \frac{1}{\varepsilon\cos\phi_R} \tag{7.20}$$

$$c_{21}B - c_{22}\varepsilon B\sin\phi_R = 0 \tag{7.21}$$

$$c_{21}B - \left(\frac{1}{\varepsilon\cos\phi_R}\right)\varepsilon B\sin\phi_R = 0 \tag{7.22}$$

$$c_{21}B - B\frac{\sin\phi_R}{\cos\phi_R} = 0 \tag{7.23}$$

$$c_{21}B - B\tan\phi_R = 0 \tag{7.24}$$

$$c_{21} = \tan\phi_R. \tag{7.25}$$

With all the elements defined we can construct the matrix C,

$$C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \tan\phi_R & \frac{1}{\varepsilon\cos\phi_R} \end{bmatrix}. \tag{7.26}$$

We can substitute this back into Equation 7.8, arriving at

$$\begin{bmatrix} S'_I \\ S'_Q \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \tan\phi_R & \frac{1}{\varepsilon\cos\phi_R} \end{bmatrix} \left(\begin{bmatrix} S_I \\ S_Q \end{bmatrix} - \begin{bmatrix} O_I \\ O_Q \end{bmatrix} \right) \tag{7.27}$$

$$\begin{bmatrix} S'_I \\ S'_Q \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \tan\phi_R & \frac{1}{\varepsilon\cos\phi_R} \end{bmatrix} \left(\begin{bmatrix} S_I - O_I \\ S_Q - O_Q \end{bmatrix} \right). \tag{7.28}$$

Solving this equation for S'_I and S'_Q yields

$$S'_I = S_I - O_I = a(t-t_0)\alpha_P\alpha_{RX}\cos[(\omega_d+\omega_e)(t-t_0)] \tag{7.29}$$

$$S'_Q = S'_I\tan\phi_R + \frac{S_Q - O_Q}{\varepsilon\cos\phi_R} = a(t-t_0)\alpha_P\alpha_{RX}\sin[(\omega_d+\omega_e)(t-t_0)]. \tag{7.30}$$

The signals S'_I and S'_Q now contain none of the non-idealities present in the original signals S_I and S_Q. Transforming S_I and S_Q into S'_I and S'_Q requires the knowledge of O_I, O_Q, ε, and ϕ_R. These four parameters are obtained from the curve fits done in the current calibration procedure. MATLAB code implementing the procedure described in this section (*Calibrate_Detailed.m*) can be found in Section A.2.3.

7.1.2 Cluttermap

A cluttermap is a dataset that contains the average clutter signal received by a radar for a given geographic location. For a stationary radar, such as the one described here, stationary ground clutter will appear at the same range with about the same signal power each time the radar scans through the azimuth it is located at. If no moving targets are within range of the radar the dataset produced by a single rotation will contain only the ground clutter. The following rotation will yield a dataset nominally identical to the first, other than noise, and by subtracting the two the result would ideally be a dataset containing nothing but zero-centered noise. However, if a moving target was present in the second scan then it's signal would be unaffected by subtraction of the cluttermap and be present in the resulting data. An example of ground clutter returns can be seen in Figure 7.1. Here the MTI has not been applied to the data so it contains a large amount of power from stationary returns which bury any returns from moving point targets such as airplanes. The cluttermap developed here operates by taking the data from a set number of scans and averaging them into a single dataset that is representative of the ground clutter returns from

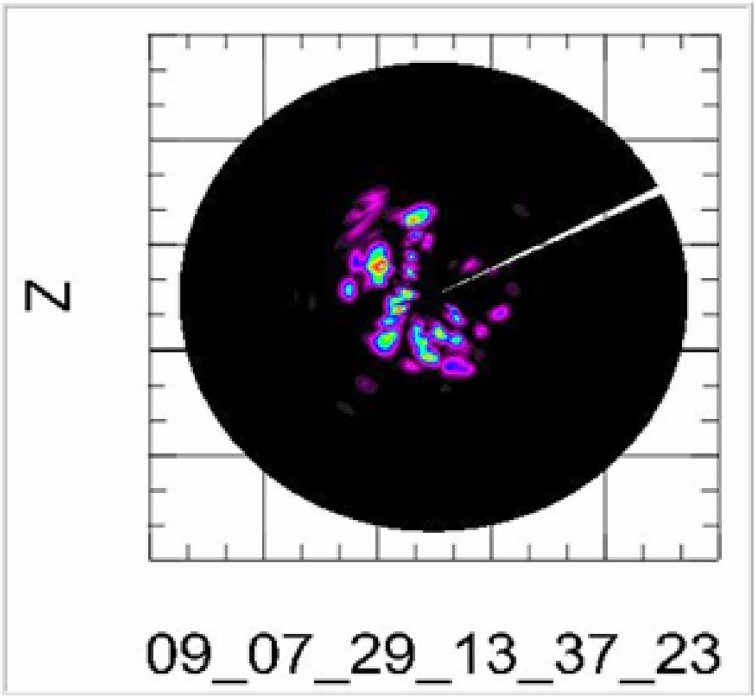

Figure 7.1. Map of ground clutter.

the surrounding area. This approach is only feasible if the datasets used to create the cluttermap contain few, or ideally zero, moving targets. In the PFRR airspace used for testing this is a valid assumption, but it may not be in more heavily used airspaces. The MATLAB code used to produce the cluttermap (*Clutter_Map.m*) can be found in Section A.2.3.

There are two foreseeable uses for the cluttermap in this system. One is as an alternative to the MTI. This could be useful if there was a need to detect slow-moving targets that would be ignored by the current system. Also, as shown in Section 5.5, the MTI (*RemoveFit.m*) uses over 50% of the time required to process a dataset. The use of a cluttermap, ignoring the generation of it, to remove ground clutter requires just a subtraction of arrays which requires orders of magnitude less processing complexity than the curve fitting done by the MTI. The second foreseeable use of a cluttermap is if the PSTAR is operating on a moving platform such as the bed of a truck. A cluttermap could be produced from each scan and changes in subsequent cluttermaps could be used to estimate the change in position and orientation of the radar.

7.2 North Alignment

The north alignment discussed in Section 4.4.1 is not satisfactory for a useful system. The north alignment is also necessary for the 2-D PSTAR currently being operated at PFRR and so more serious thought has been put into providing a valid north alignment. One approach engineers at PFRR have been investigating is the incorporation of differential GPS receiver into the PSTAR system. The GPS receiver could supply accurate information about the azimuthal orientation of the radar antenna and allow for a precise north alignment. The GPS would have the additional benefits of providing radar GPS coordinates, which are necessary to transform radar-centric target location into absolute target location, as well as providing accurate time to the system for timestamping data. The second approach being investigated by PFRR, and Prof. Hawkins of the UAF ECE department, is the creation of a parrot for the PSTAR which is a device that transmits simulated target reflections to the PSTAR. By placing this at a known bearing relative to the radar the detection of the simulated target could be used as a means of north alignment.

7.3 FPGA Based Data Collection

Much of the work presented here relies on deterministic timing. The current method of data collection involves an ADC controlled by a PXI controller, which is essentially just a standard computer running Windows XP. The complexity of the digital design in modern computers makes deterministic timing all but impossible, and the use of complex software, such as the Windows operating system, on top of the hardware further complicates the problem. While the current implementation of the 3-D PSTAR demonstrates the validity of the concept, the overhead inherent in using a Windows computer would be difficult to incorporate into a production system. An example of the problems caused by the interface to the computer during operation is the current ability to only sample data every other rotation. Keeping the computer out of the loop during operation could allow for problems like this to be solved.

One solution is to have a FPGA (Field Programmable Gate Array) be a middleman between the ADCs and the computer. The FPGA would directly control operation of the ADCs and would allow deterministic timing. The incorporation of a FPGA between the ADCs and the computer would also allow for some pre-processing to be done to the data in

real-time before it is permanently stored. This pre-processing could be used to decrease the amount of data per scan that needs to be stored for future processing, allowing for a longer duration of operation for a given data storage capacity compared to the current method of storing all raw voltage data from the ADCs. Taking the amount of pre-processing to the extreme the data stored to memory would consist of nothing but target information. This would drastically reduce the amount to data storage required and allow for real-time operation of the system. The most seamless transition of the current system to an FPGA based one would be through the use of a PXI card that contains an FPGA. Multiple companies such as National Instruments, Lyrtech, and Sundance Multiprocessor Technologies produce such cards, including versions containing high speed ADCs and DSPs.

7.4 Replace PSTAR Components

An obvious limitation to producing more than a handful of radars of the type presented here is the need for PSTAR systems to modify. While obtaining used PSTARs was possible in the numbers needed for the work presented here, it would likely become difficult to find enough used PSTARs for operational uses. One option is to purchase new PSTAR systems from Lockheed Martin, but this would add a large amount to the overall cost of the system. The PSTAR being a military radar, it may be difficult to purchase new units for legal reasons and this may also constrain the demographic of end users able to obtain the modified systems. Aside from cost and procurement issues, the PSTAR is not the ideal system for this application due to the incompleteness of documentation and certain quirks which interfere with the 3-D system, such as the DC discontinuities. For these reasons it is suggested that, if it is desired to pursue the development of an operational 3-D system, it would be best to abandon the PSTAR completely.

Replacements of the transmitter and the receiver could be likely created with COTS (Commercial Off-The-Shelf) components, but some hardware design may prove necessary. The processor module of the current PSTAR based system is responsible for things like creating the transmit pulse envelope and controlling various subsystem features during operation. If a FPGA were incorporated into the system, as discussed in Section 7.3, it could be used as the controller for the rest of the system in addition to its role in data collection. It could create the transmit pulse by keying the PXI signal generator card that is

already a part of the system for calibration. Additionally, a custom micro-controller based embedded system could be used as a real-time controller instead of an FPGA. This would add a lot of flexibility to the system by allowing for arbitrary pulse shapes and patterns to be used, expanding the market for such a system to possibly include scientific uses. The transmit frequency would also be flexible, though dependent on the transmitter and receiver hardware capabilities. This may be useful as the frequency of operation for the PSTAR lies within the military band and clearance to transmit may be difficult to obtain in some locations.

One of the largest obstacles to completely replacing the PSTAR components is the antenna. The PSTAR antenna is a high gain fan beam that, for this application, is superior to what is available on the COTS market, at least within the frequency band used by the PSTAR. To replace this would likely require a custom antenna design. If procurement is feasible, a dual PSTAR antenna such as the one used here (Section 4.2.2), perhaps along with a PSTAR pedestal, could be incorporated as the only PSTAR components of the system. Doing so would limit the frequency of operation of the system to that of the PSTAR, but it would not eliminate the benefits of replacing the receiver and transmitter.

Bibliography

[1] Raven RQ-11B Technical Specifications Document. AeroVironment Inc, www.avinc.com, 1/1/2009.

[2] RQ-4 Global Hawk Fact Sheet. Northrop Grumman, www.northrupgrumman.com, 5/2008.

[3] United States Government Accountability Office. Unmanned Aircraft Systems: Federal actions needed to ensure safety and expand their potential uses within the national airspace system. Report to congressional requestors, 5/2008.

[4] Latchman, Dr. Haniph. Brief History of UAVs. LIST Lab. University of Florida. http://aln.list.ufl.edu/uav/UAVHstry.htm, 1/17/2003.

[5] Sato, Akira. Aeronautic Operations, Yamaha Motor Co. The RMAX Helicopter UAV. 9/2/2003.

[6] Sullivan, D. V., Frost, C. R. Intelligent Mission Management for UAV Wildfire Response. American Geophysical Union, Fall Meeting 2005, abstract #IN41A-0317.

[7] Saggiana, G., et al. A UAV System for Observing Volcanoes and Natural Hazards. American Geophysical Union, Fall Meeting 2007, abstract #GC11B-05.

[8] U. S. Department of Transportation, Federal Aviation Administration. Unmanned Aircraft Systems Operations in the U. S. National Airspace System – Interim Operational Approval Guidance. AFS-400 UAS Policy 05-01. 9/16/2005.

[9] U .S. Department of Transportation, Federal Aviation Administration. Pilots' Role in Collision Avoidance. 90-48C. 3/18/1983.

[10] U. S. Department of Transportation, Federal Aviation Administration. Airworthiness Certification of Aircraft and Related Products. National Policy. 4/18/2007.

[11] U. S. Department of Transportation, Federal Aviation Administration. Unmanned Aircraft Systems (UAS) Fact Sheet. 11/12/2009.

[12] Technical Manual Operator's Manual. Radar Set AN/PPQ-2 (NSN 1430-01-347-7673). Department of the Army. 9/1993.

[13] Gruszczynski, Jerzy. "PSTAR radar sets for Switzerland." Journal of Electronic Defense 2/1/2003.

[14] PSTAR Portable Search and Target Acquisition Radar fact sheet. Lockheed Martin. 7/2003.

[15] Technical Manual Unit Maintenance Manual Including Repair Parts and Special Tools List (RPSTL). Radar Set AN/PPQ-2 (NSN 1430-01-347-7673). Department of the Army. 9/1993.

[16] Breit, G. and M. A. Tuve. 1926. A Test of the Existence of the Conducting Layer. Phys. Rev. 28:554-575.

[17] Peebles, Peyton Z. Radar Principles. Wiley & Sons, Inc., 1998.

[18] Webster, John G. The Measurement, Instrumentation, and Sensors Handbook. CRC Press, LLC. 1999.

[19] MATLAB Central File Exchange.
http://www.mathworks.com/matlabcentral/fileexchange/.

[20] Richards, Mark A. Fundamentals of Radar Signal Processing. McGraw-Hill Companies, Inc. 2005.

Appendix A
Appendix

A.1 Data Collection Code (LabVIEW)

A.1.1 Top Level Functions

Record_Data_5deg_128.vi

This function continuously collects data in bursts of 128 pulses every 5°.

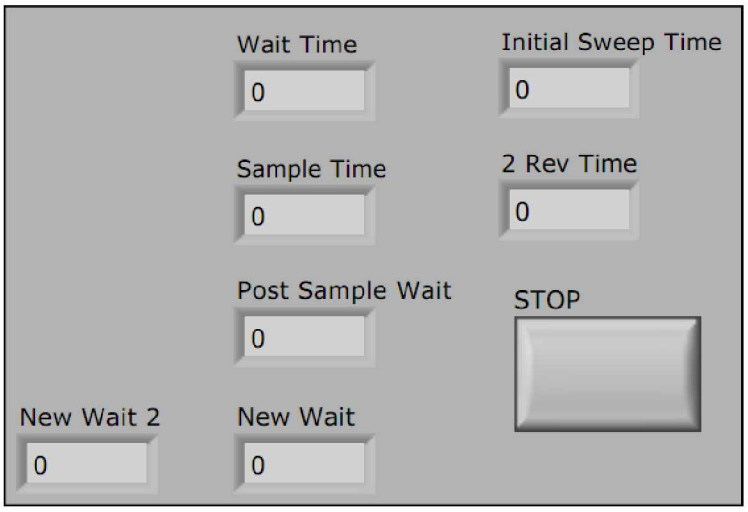

Figure A.1. Front panel of Record_Data_5deg_128.vi.

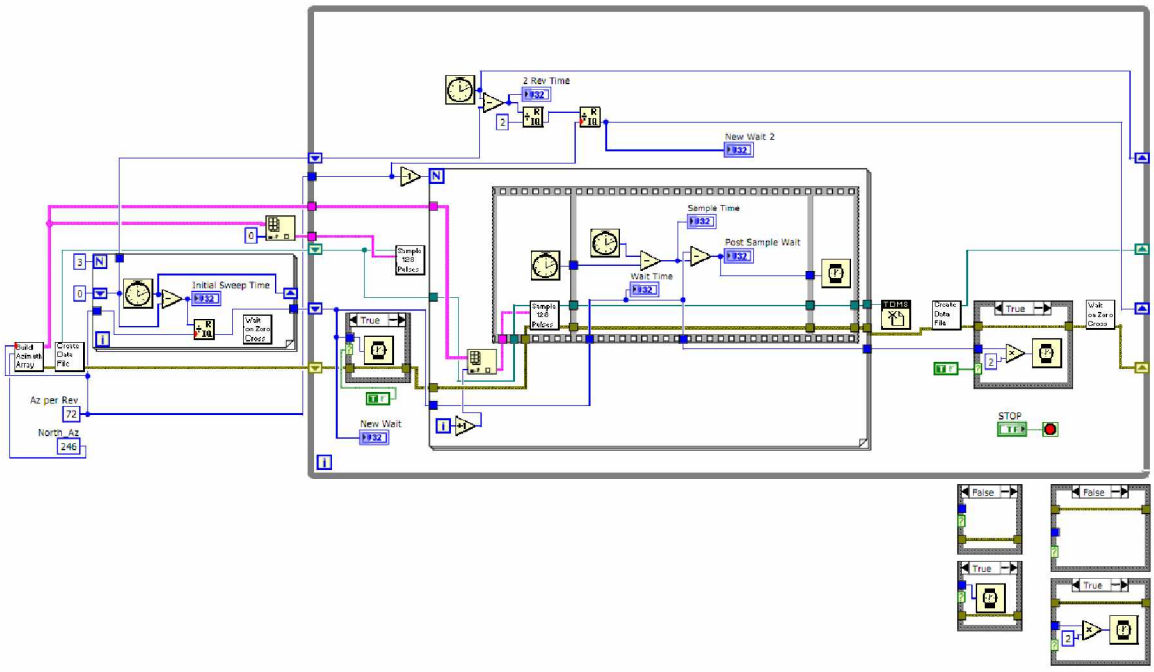

Figure A.2. Block diagram of Record_Data_5deg_128.vi.

Record_Data_10deg_256.vi

This function continuously collects data in bursts of 256 pulses every 10°.

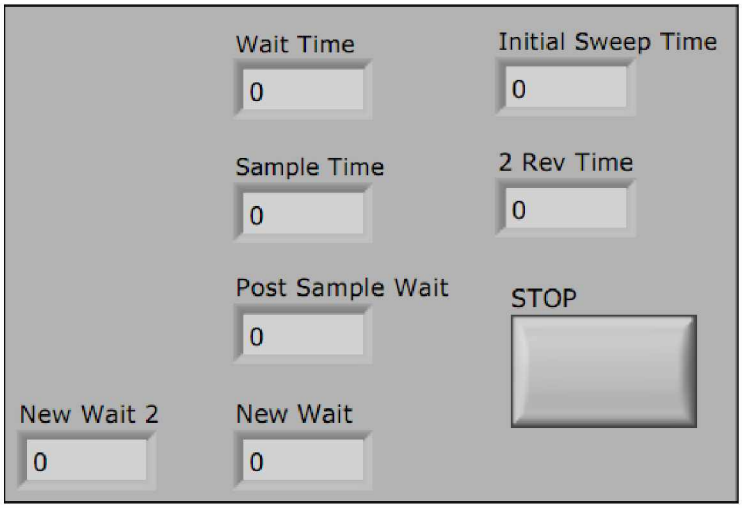

Figure A.3. Front panel of Record_Data_5deg_128.vi.

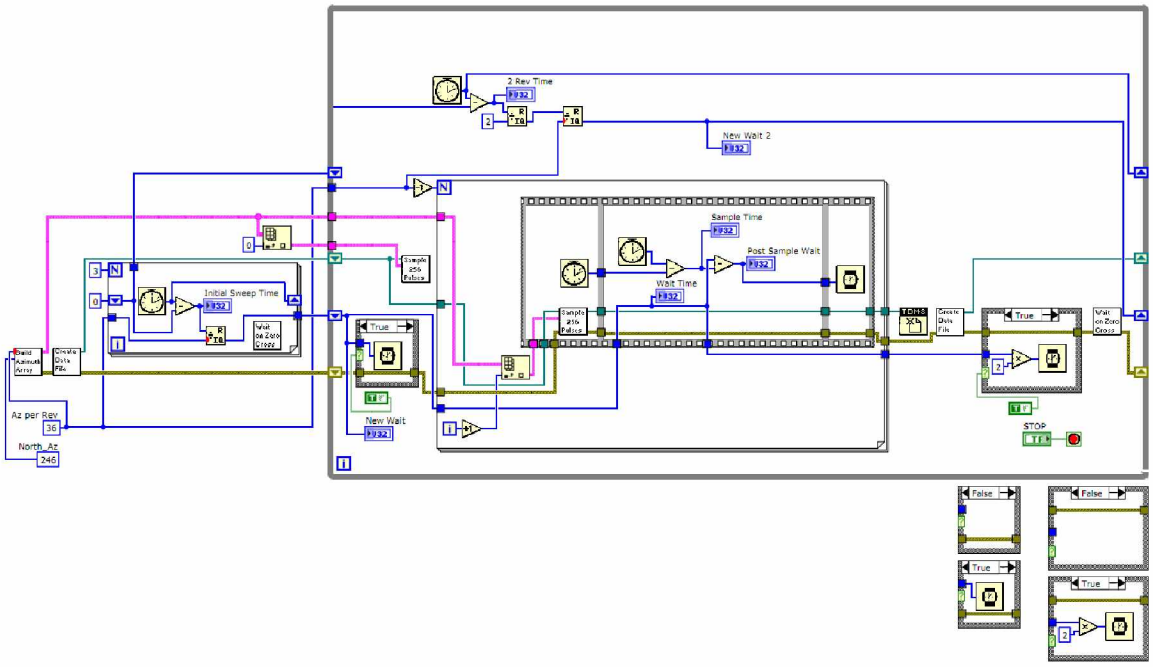

Figure A.4. Block diagram of Record_Data_5deg_128.vi.

Startup_Cal.vi

This function continuously collects used in the startup calibration.

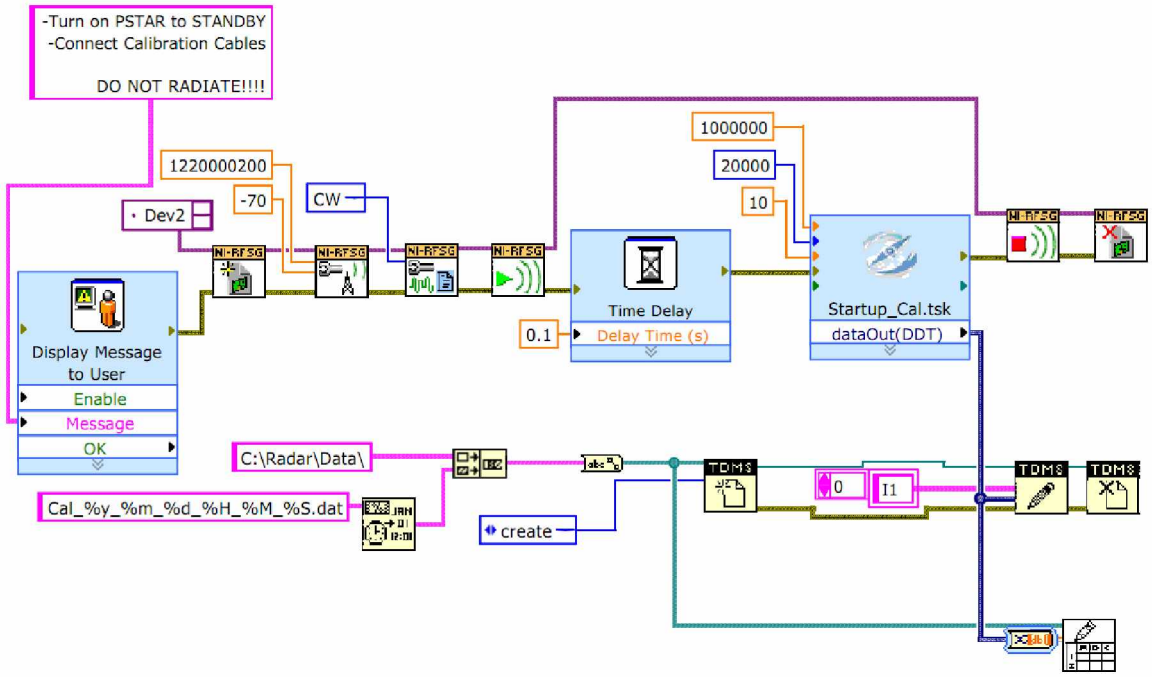

Figure A.5. Block diagram of Startup_Cal.vi.

A.1.2 Called Functions

Az_North_Shift.vi

This function corrects for the north alignment.

Figure A.6. Function block of Record_Data_5deg_128.vi.

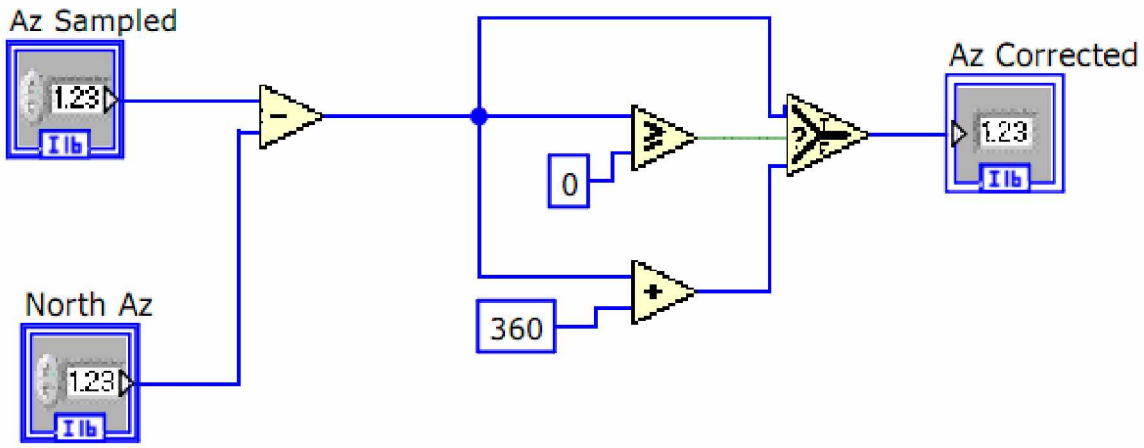

Figure A.7. Block diagram of Az_North_Shift.vi.

Build_Az_Array.vi

This function builds an array containing all azimuths to be sampled.

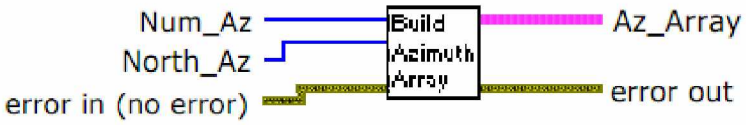

Figure A.8. Function block of Build_Az_Array.vi.

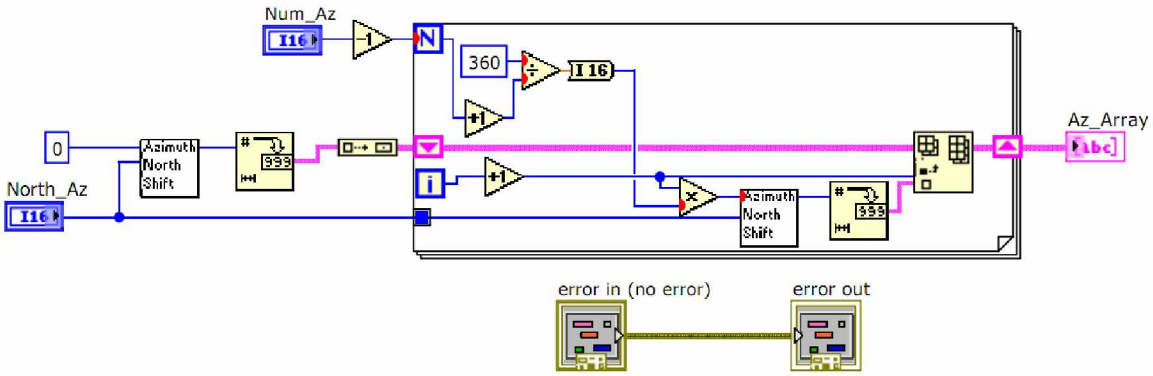

Figure A.9. Block diagram of Build_Az_Array.vi.

Create_Data_File.vi

This function creates a TDMS data file.

Figure A.10. Function block of Create_Data_File.vi.

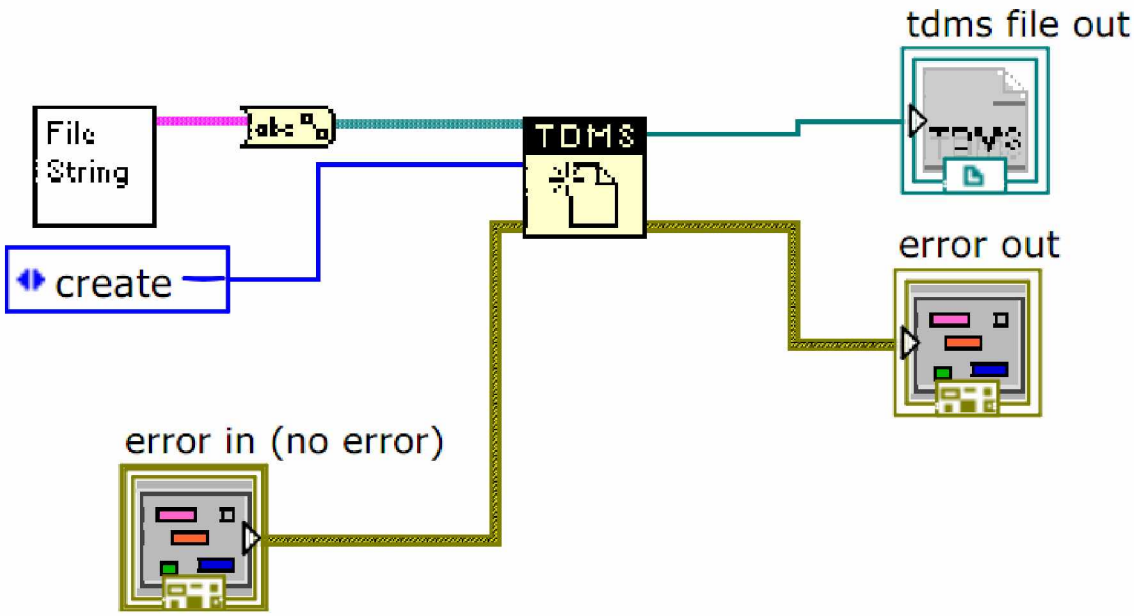

Figure A.11. Block diagram of Create_Data_File.vi.

File_String.vi

This function creates a file string containing the current date and time.

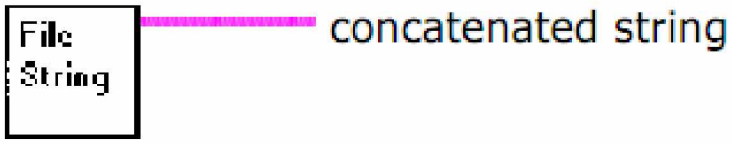

Figure A.12. Function block of File_String.vi.

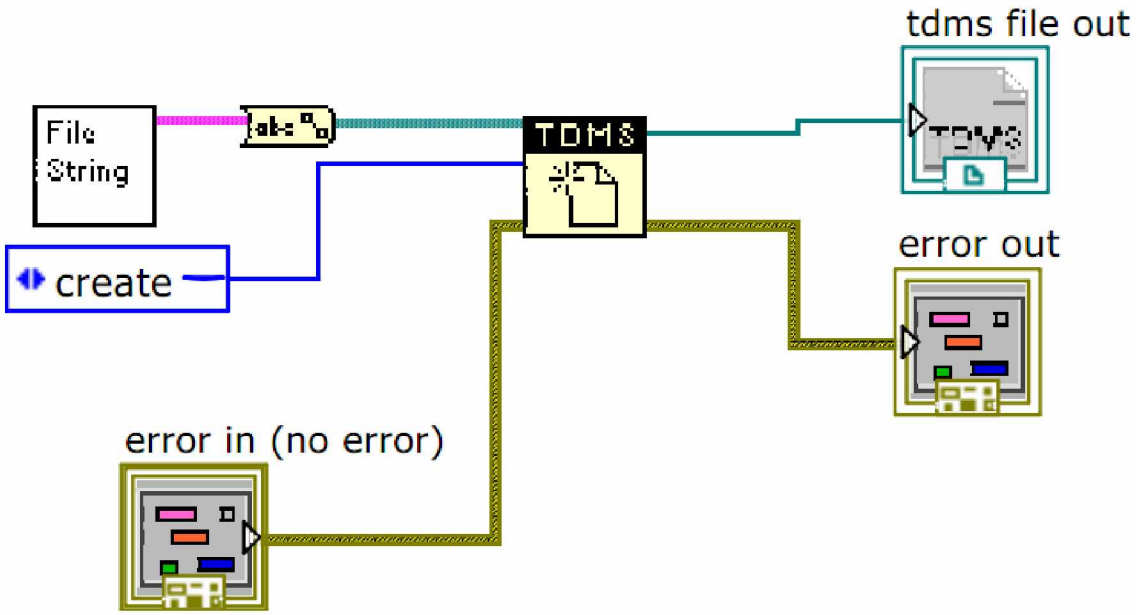

Figure A.13. Block diagram of File_String.vi.

Sample_128_Pulses.vi

This function samples 128 successive pulses.

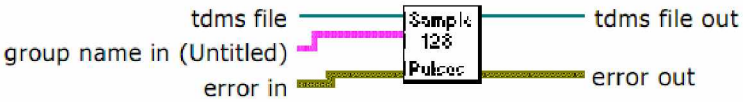

Figure A.14. Function Sample_128_Pulses.vi.

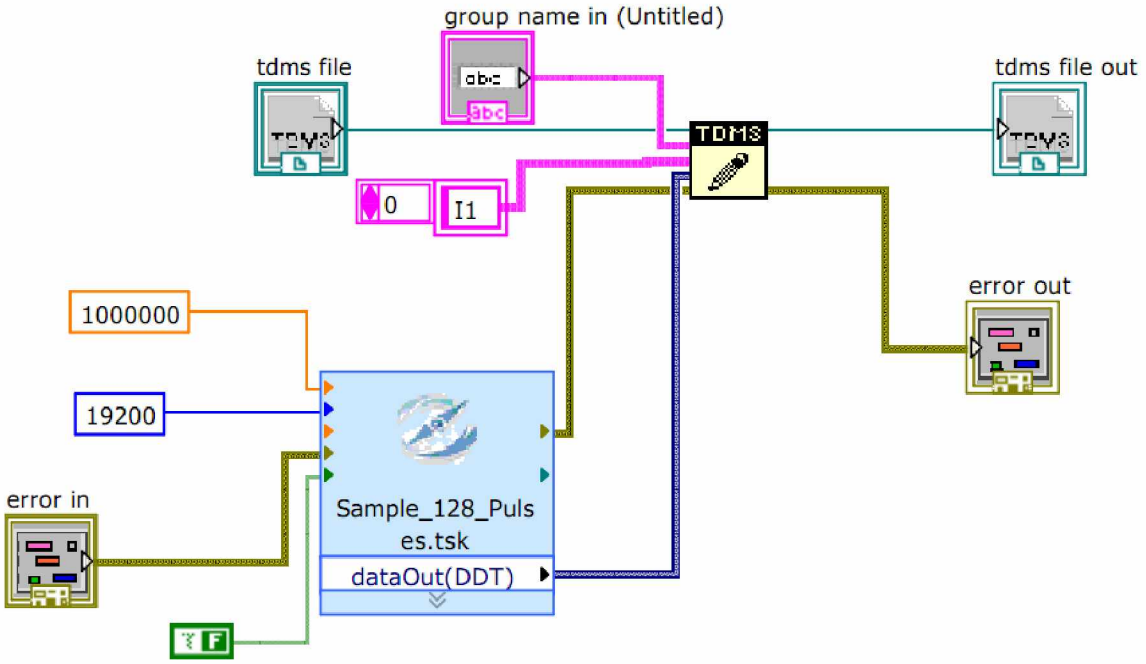

Figure A.15. Block diagram of Sample_128_Pulses.vi.

Sample_256_Pulses.vi

This function samples 256 successive pulses.

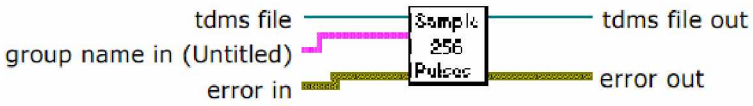

Figure A.16. Function Sample_256_Pulses.vi.

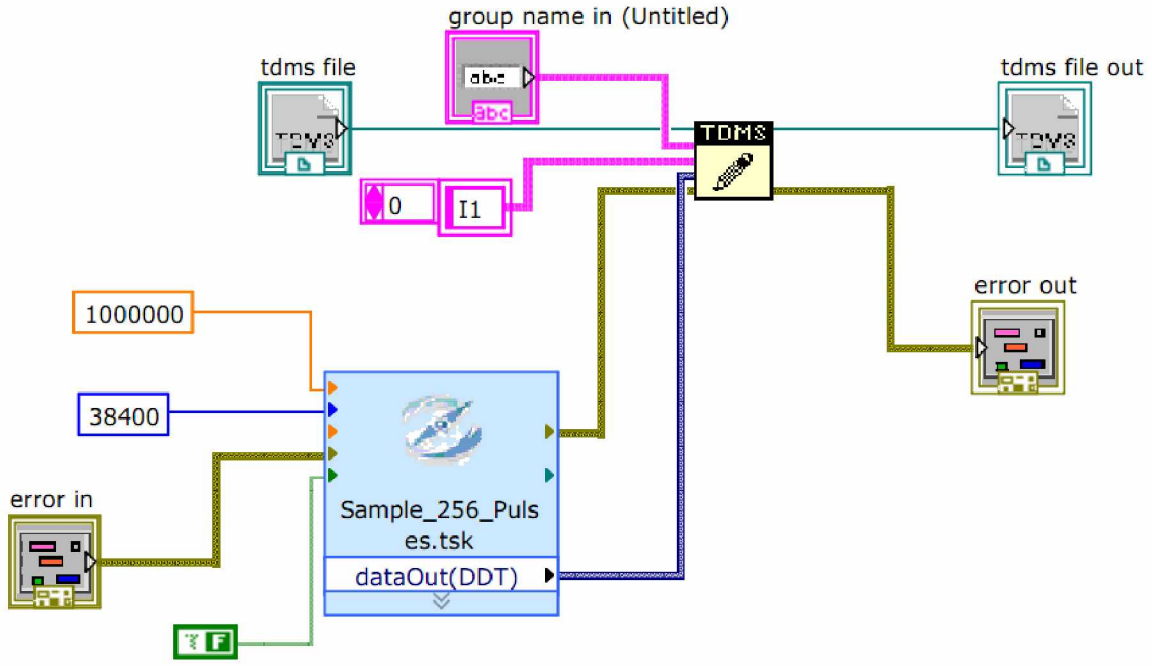

Figure A.17. Block diagram of Sample_256_Pulses.vi.

Wait_on_Zero_Cross.vi

This function waits for the next pedestal zero crossing.

Figure A.18. Wait_on_Zero_Cross.vi.

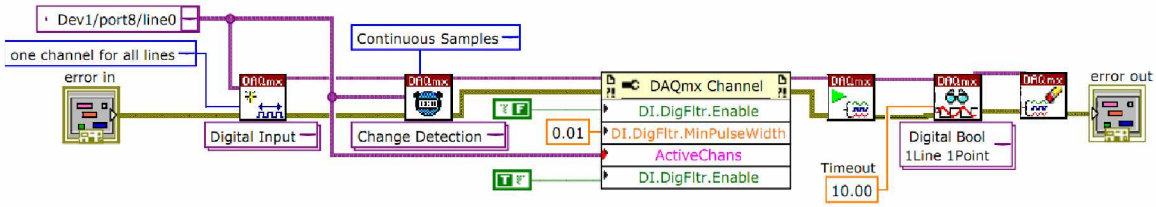

Figure A.19. Block diagram of Wait_on_Zero_Cross.vi.

A.2 Data Processing Code (MATLAB)

A.2.1 Top Level Functions

Run_All_Folder_3D.m

This function processes all TDMS files contained within a folder with the aid of a calibration datafile.

```
%Process Folder of .tdms Files

clear
clc

%Define Constants
Fc = 1.22e9;                    %TX Carrier Freq
Npulses = 256;                  %Number of Pulses per Az
Nsamples = 150;                 %Number of Samples per Pulse
ts = 1e-6;                      %Sampling Time (s)
tau = 8e-6;                     %Pulse Width (s)
PRF = 6250;                     %Pulse Repitition Freq (Hz)
Min_Dopp = 25;                  %Minimum Doppler Velocity (+-m/s)
d = 0.48;                       %Antenna Spacing (m)
PolyDeg = 2;                    %Polynomial Fit Degree
DiscThresh = 1.2;               %Discontinuity Threshold
TriFiltLength = 5;              %Length of Triangle Filter
TriFiltWeight = 0.9;            %Weight of Triangle Filter
DoppThresh = 10;                %Doppler Threshold

IPP = 1/PRF;                            %Inter Pulse Period (s)
Pulse = zeros(1,150);
Pl = round(tau/ts);                     %TX Pulse length in pts
Pulse(1:Pl) = 1;                        %TX Pulse
t = tau:ts:(Nsamples-1)*ts+tau;         %Time Vector (s)
```

```
Ranges = t.*((3e8)/2);                  %Range Vector (m)
f = linspace(-PRF/2,PRF/2,Npulses);     %Freq Vector (Hz)
v = -f.*(3e8)/(2*Fc);                   %Velocity Vector (m/s)
vb = find(v>-Min_Dopp & v<Min_Dopp);    %Velocity indexes to blank
vb(1) = [];                             %These ignored later
vb(1) = [];
vb(length(vb)) = [];
vb(length(vb)) = [];
l = (3e8)/Fc;                             %Wavelength (m)

%Get Data Folder
Data_Folder = uigetdir('C:\','Select Data Folder');
Data_File_List = dir([Data_Folder '\*.tdms']);

%Get Channel Phase Difference
PhiDiff = Calibrate();

FolderWait = waitbar(0,'Data Folder');
FileWait = waitbar(0,'Data File');

Targetsf = [];
Tnum = 1;

RangeAzT = [];
RangeAzTf = [];

TS = 0;

colors = colormap(jet(length(Data_File_List)));

for df = 1:length(Data_File_List)
```

```
    waitbar(df/length(Data_File_List),FolderWait);

    [I1a,Q1a,I2a,Q2a,Az,Time] =
     LoadTDMS([Data_Folder '\' Data_File_List(df).name]);

    if isempty(RangeAzT)
        RangeAzT = zeros(size(Az(:,1)),Nsamples);
        RangeAzTf = zeros(size(Az(:,1)),Nsamples);
    end

    Targets = [];

    if df == 1
        T0 = Time;
    end

    for a = 1:length(Az(:,1))
%     for a = 25:25
        waitbar(a/length(Az(:,1)),FileWait);

        for i = 1:Npulses
            I1(i,:) = I1a(1+Nsamples*(i-1):Nsamples*i,a);
            Q1(i,:) = Q1a(1+Nsamples*(i-1):Nsamples*i,a);
            I2(i,:) = I2a(1+Nsamples*(i-1):Nsamples*i,a);
            Q2(i,:) = Q2a(1+Nsamples*(i-1):Nsamples*i,a);
        end

        DI1 = Find_Discontinuity(I1,Nsamples,Npulses,DiscThresh);
        DQ1 = Find_Discontinuity(Q1,Nsamples,Npulses,DiscThresh);
        DI2 = Find_Discontinuity(I2,Nsamples,Npulses,DiscThresh);
        DQ2 = Find_Discontinuity(Q2,Nsamples,Npulses,DiscThresh);
```

```
Discs = unique([DI1 DQ1 DI2 DQ2]);

I1 =
 Fix_Discontinuity(I1,Nsamples,Npulses,DiscThresh,Discs);
Q1 =
 Fix_Discontinuity(Q1,Nsamples,Npulses,DiscThresh,Discs);
I2 =
 Fix_Discontinuity(I2,Nsamples,Npulses,DiscThresh,Discs);
Q2 =
 Fix_Discontinuity(Q2,Nsamples,Npulses,DiscThresh,Discs);

[S1,S2] = Correlate(I1,Q1,I2,Q2,Pulse);

[S1f] = RemoveFit(S1,PolyDeg,Npulses,Nsamples);
[S2f] = RemoveFit(S2,PolyDeg,Npulses,Nsamples);

M = sum(abs(S1f.*S2f));
M1 = sum(abs(S1f));
M2 = sum(abs(S2f));

Mf = TriFilter(M,Pl,TriFiltLength,TriFiltWeight);

P = find(Mf > 0);

if length(P) >= 2
    for i = 1:length(P)-1
        if P(i)+1 == P(i+1)
            if Mf(P(i)) > Mf(P(i+1))
                P(i+1) = 0;
            else
```

```
                P(i) = 0;
            end
        end
    end

    P(P == 0) = [];
end

RangeAz(a,:) = M;
RangeAzf(a,:) = Mf;

if ~isempty(Targets)
    PTL = length(Targets(:,1));
else
    PTL = 0;
end

if ~isempty(P)
    [D1] = Target_Dopplers(S1f,P,Npulses,vb);
    [D2] = Target_Dopplers(S2f,P,Npulses,vb);

    D = D1+D2;

    D(vb(1)-1,:) = D(vb(1));
    D(vb(length(vb))+1,:) =  D(vb(length(vb)),:);

    TL = Doppler_Detect(D,vb,DoppThresh,v);

    TR = Ranges(P);

    [dPhi] = PhaseDifference(S1f,S2f,P,PhiDiff);
```

```
            ElevAngle = Elevation_Angle(d,l,dPhi(:,1));

            dfn = Data_File_List(df).name(1:17);

            for i = PTL+1:PTL+length(TR)
                Targets(i,1) = a;                       %Azimuth
                Targets(i,2) = P(i-PTL);                %Range
                Targets(i,3) = TL(i-PTL);               %Doppler
                Targets(i,4) = ElevAngle(i-PTL,1);      %Phase Diff
                Targets(i,5) = Az(a,2);                 %Time
                Targets(i,6) = str2num(char([dfn(10:11)
                 dfn(13:14) dfn(16:17)]));              %Date

                Targets(i,7) = dPhi(i-PTL,2);
                Targets(i,8:7+length(ElevAngle(1,:))) =
                 ElevAngle(i-PTL,:);
                Targets(i,10) = M(P(i-PTL));
                Targets(i,11) = Mf(P(i-PTL));
            end
        end

        clear D P dPhi;
    end

    if ~isempty(Targets)
        [Targetsf] =
         BuildTargets2(Targets,Targetsf,Az,Ranges,v,RangeAzf);

        if ~isempty(Targetsf)
            polarmy(0,0,' ',25000,5);
```

```
            hold on;
            title([Time(10:11) ':' Time(13:14) ':' Time(16:17)
             '    ' Time(4:5) '/' Time(7:8) '/20' Time(1:2)]);

            SlantRange = Targetsf(TS+1:length(Targetsf(:,1)),2);
            GroundRange = SlantRange.*
             cosd(Targetsf(TS+1:length(Targetsf(:,1)),4));
            Altitude = SlantRange.*
             sind(Targetsf(TS+1:length(Targetsf(:,1)),4));

            txtX = SlantRange.*cos(
             deg2rad(Targetsf(TS+1:length(Targetsf(:,1)),1)));
            txtY = SlantRange.*sin(
             deg2rad(Targetsf(TS+1:length(Targetsf(:,1)),1)));
            text(txtX,txtY,'x','color',colors(df,:));

            TS = length(Targetsf(:,1));

            RangeAzT = RangeAzT + RangeAz;
            RangeAzTf = RangeAzTf + RangeAzf;

            clear SlantRange GroundRange Altitude txtX txtY
        end

        if df == 1
            T0 = Time;
        end
    end
end

colorbar('YTickLabel',{[T0(10:11) ':' T0(13:14) ':' T0(16:17)],
```

```
 ' ',' ',' ',' ',' ',[Time(10:11) ':' Time(13:14) ':'
  Time(16:17)]});

close(FileWait);
close(FolderWait);
```

Run_All_Sim_Data.m

This function processes a simulated data file created by the function Sim_Data.m.

```
%Process Folder of .tdms Files

clear
clc

%Define Constants
Fc = 1.22e9;                    %TX Carrier Freq
Npulses = 128;                  %Number of Pulses per Az
Nsamples = 150;                 %Number of Samples per Pulse
ts = 1e-6;                      %Sampling Time (s)
tau = 8e-6;                     %Pulse Width (s)
PRF = 6250;                     %Pulse Repitition Freq (Hz)
Min_Dopp = 25;                  %Minimum Doppler Velocity (+-m/s)
d = 0.48;                       %Antenna Spacing (m)
PolyDeg = 2;                    %Polynomial Fit Degree
DiscThresh = 1.2;               %Discontinuity Threshold
TriFiltLength = 5;              %Length of Triangle Filter
TriFiltWeight = 0.9;            %Weight of Triangle Filter
% TriFiltLength = 8;            %Length of Triangle Filter
% TriFiltWeight = 1;            %Weight of Triangle Filter
DoppThresh = 10;                %Doppler Threshold
```

```
IPP = 1/PRF;                              %Inter Pulse Period (s)
Pulse = zeros(1,150);
Pl = round(tau/ts);                       %TX Pulse length in pts
Pulse(1:Pl) = 1;                          %TX Pulse
t = tau:ts:(Nsamples-1)*ts+tau;           %Time Vector (s)
Ranges = t.*((3e8)/2);                    %Range Vector (m)
f = linspace(-PRF/2,PRF/2,Npulses);       %Freq Vector (Hz)
v = -f.*(3e8)/(2*Fc);                     %Velocity Vector (m/s)
vb = find(v>-Min_Dopp & v<Min_Dopp);      %Velocity indexes to blank
vb(1) = [];                               %These ignored later
vb(1) = [];
vb(length(vb)) = [];
vb(length(vb)) = [];
l = (3e8)/Fc;                             %Wavelength (m)

PhiDiff = 0;

load(uigetfile('*.mat','Select Sim Data File'));

Targetsf = [];
Tnum = 1;

TS = 0;

I1a = data(1,:)';
Q1a = data(2,:)';
I2a = data(3,:)';
Q2a = data(4,:)';

Targets = [];
```

```
df = 1;
a = 1;

for i = 1:Npulses
    I1(i,:) = I1a(1+Nsamples*(i-1):Nsamples*i,a);
    Q1(i,:) = Q1a(1+Nsamples*(i-1):Nsamples*i,a);
    I2(i,:) = I2a(1+Nsamples*(i-1):Nsamples*i,a);
    Q2(i,:) = Q2a(1+Nsamples*(i-1):Nsamples*i,a);
end

DI1 = Find_Discontinuity(I1,Nsamples,Npulses,DiscThresh);
DQ1 = Find_Discontinuity(Q1,Nsamples,Npulses,DiscThresh);
DI2 = Find_Discontinuity(I2,Nsamples,Npulses,DiscThresh);
DQ2 = Find_Discontinuity(Q2,Nsamples,Npulses,DiscThresh);

Discs = unique([DI1 DQ1 DI2 DQ2]);

I1 = Fix_Discontinuity(I1,Nsamples,Npulses,DiscThresh,Discs);
Q1 = Fix_Discontinuity(Q1,Nsamples,Npulses,DiscThresh,Discs);
I2 = Fix_Discontinuity(I2,Nsamples,Npulses,DiscThresh,Discs);
Q2 = Fix_Discontinuity(Q2,Nsamples,Npulses,DiscThresh,Discs);

[S1,S2] = Correlate(I1,Q1,I2,Q2,Pulse);

[S1f] = RemoveFit(S1,PolyDeg,Npulses,Nsamples);
[S2f] = RemoveFit(S2,PolyDeg,Npulses,Nsamples);

M = sum(abs(S1f.*S2f));
M1 = sum(abs(S1f));
M2 = sum(abs(S2f));
```

```
Mf = TriFilter(M,Pl,TriFiltLength,TriFiltWeight);

P = find(Mf > 0);

if length(P) >= 2
    for i = 1:length(P)-1
        if P(i)+1 == P(i+1)
            if Mf(P(i)) > Mf(P(i+1))
                P(i+1) = 0;
            else
                P(i) = 0;
            end
        end
    end

    P(P == 0) = [];
end

RangeAz(a,:) = M;
RangeAzf(a,:) = Mf;

if ~isempty(Targets)
    PTL = length(Targets(:,1));
else
    PTL = 0;
end

if ~isempty(P)
    [D1] = Target_Dopplers(S1f,P,Npulses,vb);
    [D2] = Target_Dopplers(S2f,P,Npulses,vb);
```

```
    D = D1+D2;

    D(vb(1)-1,:) = D(vb(1));
    D(vb(length(vb))+1,:) =  D(vb(length(vb)),:);

    TL = Doppler_Detect(D,vb,DoppThresh,v);

    TR = Ranges(P);

    [dPhi] = PhaseDifference(S1f,S2f,P,PhiDiff);

    ElevAngle = Elevation_Angle(d,l,dPhi(:,1));

    for i = PTL+1:PTL+length(TR)
        Targets(i,1) = a;                              %Azimuth
        Targets(i,2) = P(i-PTL);                       %Range
        Targets(i,3) = TL(i-PTL);                      %Doppler
        Targets(i,4) = ElevAngle(i-PTL,1);             %Phase Diff
        Targets(i,5) = -1;                             %Time
        Targets(i,6) = -1;                             %Date

        Targets(i,7) = dPhi(i-PTL,2);
        Targets(i,8:7+length(ElevAngle(1,:))) =
         ElevAngle(i-PTL,:);
        Targets(i,10) = M(P(i-PTL));
        Targets(i,11) = Mf(P(i-PTL));
    end
end

if ~isempty(Targets)
    TT = Targets(Targets(:,3) ~= 0,:);
```

```
if ~isempty(Targetsf)
    ST = length(Targetsf(:,1));
else
    ST = 0;
end

CT = 1;

for i = 1:length(Targets(:,1))
    if Targets(i,3) ~= 0              %Don't plot v=0 targets
        h = -1;
        m = -1;
        s = -1;

        Targetsf(ST+CT,1) = -1;
        Targetsf(ST+CT,2) = Ranges(Targets(i,2));

        if Targets(i,3) ~= 0
            Targetsf(ST+CT,3) = v(Targets(i,3));
        else
            Targetsf(ST+CT,3) = 0;
        end

        Targetsf(ST+CT,4) = Targets(i,4);
        Targetsf(ST+CT,5) = h;
        Targetsf(ST+CT,6) = m;
        Targetsf(ST+CT,7) = s;

        CT = CT+1;
    end
```

```
    end
end
```

Sim_Data.m

This function creates a simulated data file.

```
%Simulated PSTAR Data

clear all

%PSTAR Parameters
c = 3e8;                            %Speed of light [m/s]
T = 7e-6;                           %TX Pulse Length [s]
IPP = 160e-6;                       %Interpulse Period [s]
f = 1.22e9;                         %TX Frequency [Hz]
Np = 128;                           %Number of Pulses
Tsim = 1e-10;                       %Sim Data Sample Time [s]
t = [Tsim:Tsim:IPP*Np];             %Sim Data Time Vector  [s]
Ts = 10^-6;                         %Baseband Sample Time [s]
Ptx = 1000;                         %TX Power [W]
Gant = 10^(19.5/10);                %Antenna Gain [Linear]
GrxMain = 10^(78/10);               %Main RX Power Gain [Linear]
GrxSLC = 10^(83/10);                %SLC RX Power Gain [Linear]
d = 0.48;                           %Antenna Spacing [m]

Atx = sqrt(2*Ptx);                  %TX Amplitude (R=1) [V]
IPPn = IPP/Tsim;                    %Samples per IPP
Tn = T/Tsim;                        %Samples per Pulse
deci = Ts/Tsim;                     %Demod Decimation Facotor
l = (3e8)/f;                        %Wavelength [m]
```

```
Tdist = [10050];                    %Target Distance [m]
Trcs = [1];                         %Target RCS [m^2]
Tvel = [50];                        %Target Radial Velocity [m/s]
Tel = [5];                          %Target Elevation Angle [deg]

Tdelay = 2*Tdist/c;                 %Target Reflection Delay [s]
Tdeln = Tdelay/Tsim;                %Samples per Delay
Tdop = -2*f*Tvel/c;                 %Target Doppler Frequency [Hz]

Prx = 1e-11;
Arx = sqrt(2*Prx);                  %RX Amplitude (R=1) [V]
PiqMain = Prx*GrxMain;
PiqSLC = Prx*GrxSLC;
AiqMain = sqrt(2*PiqMain);
AiqSLC = sqrt(2*PiqSLC);
ap = Arx/Atx;                       %Transmission Loss [Linear]
arMain = AiqMain/Arx;               %Receiver Voltage Gain [Linear]
arSLC = AiqSLC/Arx;
Tph = 2*pi*(d/l)*sind(Tel);         %Target Phase Difference [rad]

%Receiver Parameters
OI1 = 0;                            %I1 Channel DC Offset [V]
OQ1 = 0;                            %Q1 Channel DC Offset [V]
OI2 = 0;                            %I1 Channel DC Offset [V]
OQ2 = 0;                            %Q1 Channel DC Offset [V]
sc1 = 1;                            %Ch 1 IQ Amp Scale
sc2 = 1;                            %Ch 2 IQ Amp Scale
CHsc = 1;                           %Channel Amp Scale
Ph1 = 0;                            %Ch 1 IQ Phase Offset [rad]
Ph2 = 0;                            %Ch 2 IQ Phase Offset [rad]
dPhiCH = 0;                         %Ch Phase Offset      [rad]
```

```
%Noise
NpwrMain = 3.3e-6;                %Main Noise Power [W]
NpwrSLC = 6.36*NpwrMain;          %SLC Noise Power [W]
Nph = 0;                          %Target Phase Jitter [0:1]

NvarMain = (NpwrMain)^0.25;       %Noise Variance [V]
NvarSLC = (NpwrSLC)^0.25;         %Noise Variance [V]

for m = 1:Np
    S1rx = zeros(1,IPPn);
    S2rx = zeros(1,IPPn);

    for i = 1:length(Tdist)
        Tspan1 = round(Tdeln(i):Tdeln(i)+Tn-1);
        Tspan2 =
         round(Tdeln(i)+(m-1)*IPPn:Tdeln(i)+Tn-1+(m-1)*IPPn);
        Nphase = Nph*2*pi*randn(1,length(Tspan1));
        S1rx(Tspan1) = S1rx(Tspan1)+ap(i)*arMain(i)*Atx*
         sin(2*pi*(f-Tdop(i))*t(Tspan2)+Nphase);
        S2rx(Tspan1) = S2rx(Tspan1)+ap(i)*arSLC(i)*Atx*
         sin(2*pi*(f-Tdop(i))*t(Tspan2)+Tph(i)+Nphase);
    end

    NoiseMain = NvarMain*randn(1,length(S1rx));
    S1rx = S1rx + NoiseMain;
    NoiseSLC = NvarSLC*randn(1,length(S2rx));
    S2rx = S2rx + NoiseSLC;

    Tspan = round((m-1)*IPPn+1:m*IPPn);
    LO1i = cos(2*pi*f*t(Tspan));
```

```
    LO1q = sin(2*pi*f*t(Tspan)-Ph1);
    LO2i = cos(2*pi*f*t(Tspan)-dPhiCH);
    LO2q = sin(2*pi*f*t(Tspan)-Ph2-dPhiCH);
    I1rx = S1rx.*LO1i;
    Q1rx = S1rx.*LO1q;
    I2rx = S2rx.*LO2i;
    Q2rx = S2rx.*LO2q;

    SI1rx(m,:) = decimate(I1rx,deci);
    SQ1rx(m,:) = decimate(Q1rx,deci);
    SI2rx(m,:) = decimate(I2rx,deci);
    SQ2rx(m,:) = decimate(Q2rx,deci);
end

SI1 = CHsc*sc1*SI1rx(1,[7:156])+OI1;
SQ1 = CHsc*SQ1rx(1,[7:156])+OQ1;
SI2 = sc2*SI2rx(1,[7:156])+OI2;
SQ2 = SQ2rx(1,[7:156])+OQ2;

for i = 2:Np
    SI1 = [SI1 CHsc*sc1*SI1rx(i,[7:156])+OI1];
    SQ1 = [SQ1 CHsc*SQ1rx(i,[7:156])+OQ1];
    SI2 = [SI2 sc2*SI2rx(i,[7:156])+OI2];
    SQ2 = [SQ2 SQ2rx(i,[7:156])+OQ2];
end

data(1,:) = SI1;
data(2,:) = SQ1;
data(3,:) = SI2;
data(4,:) = SQ2;
```

```
save simdata.mat data;
```

A.2.2 Called Functions

BuildTargets2.m

This function takes valid targets from the temporary variable Targets and stores them in the permanent variable Targetsf.

```
function [Targetsf]
 = BuildTargets2(Targets,Targetsf,Az,Ranges,v,RangeAzf)

%Add Targets to Targetsf

TT = Targets(Targets(:,3) ~= 0,:);

if ~isempty(Targetsf)
    ST = length(Targetsf(:,1));
else
    ST = 0;
end

CT = 1;

for i = 1:length(Targets(:,1))
    if Targets(i,3) ~= 0                    %Don't plot v=0 targets
        h = hour(Targets(i,5));
        m = minute(Targets(i,5));
        s = second(Targets(i,5));

        Targetsf(ST+CT,1) = Az(Targets(i,1),1);
        Targetsf(ST+CT,2) = Ranges(Targets(i,2));
```

```
        if Targets(i,3) ~= 0
            Targetsf(ST+CT,3) = v(Targets(i,3));
        else
            Targetsf(ST+CT,3) = 0;
        end

        Targetsf(ST+CT,4) = Targets(i,4);
        Targetsf(ST+CT,5) = h;
        Targetsf(ST+CT,6) = m;
        Targetsf(ST+CT,7) = s;

        CT = CT+1;
    end
end
```

Calibrate.m

This function determines the channel phase difference using a calibration datafile.

```
function [dPhiCH] = Calibrate()

%Extract Cal Parameters

[Cal_File,Cal_Folder] =
 uigetfile('*.dat','Select Calibration File');

CF = [Cal_Folder Cal_File];

data = load(CF)';

I1 = data(:,1);
Q1 = data(:,2);
```

```
I2 = data(:,3);
Q2 = data(:,4);

[tI1,I1f] = Remove_RXBlanking(I1);
[tQ1,Q1f] = Remove_RXBlanking(Q1);
[tI2,I2f] = Remove_RXBlanking(I2);
[tQ2,Q2f] = Remove_RXBlanking(Q2);

I1d = mean(I1);
Q1d = mean(Q1);
I2d = mean(I2);
Q2d = mean(Q2);

I1f = I1f-I1d;
Q1f = Q1f-Q1d;
I2f = I2f-I2d;
Q2f = Q2f-Q2d;

I1ffit = fit(tI1',I1f','sin1');
Q1ffit = fit(tQ1',Q1f','sin1');
I2ffit = fit(tI2',I2f','sin1');
Q2ffit = fit(tQ2',Q2f','sin1');

dPhiCH = rad2deg((I1ffit.c1-I2ffit.c1+Q1ffit.c1-Q2ffit.c1)/2);
```

convertTDMS.m

This function loads TDMS files into MATLAB. It was not written by the author of this thesis. It can be found on the MATLAB Central File Exchange [18].

Correlate.m

This function correlates the received signal with the transmit pulse.

```
function [S1,S2] = Correlate(I1,Q1,I2,Q2,Pulse)

%Correlates RX signal with TX Pulse

Npulses = length(I1(:,1));
Nsamples = length(I1(1,:));

for i = 1:Npulses
    CI1(i,:) = xcorr(I1(i,:),Pulse);
    CQ1(i,:) = xcorr(Q1(i,:),Pulse);
    CI2(i,:) = xcorr(I2(i,:),Pulse);
    CQ2(i,:) = xcorr(Q2(i,:),Pulse);
end

CI1(:,1:Nsamples-1) = [];
CQ1(:,1:Nsamples-1) = [];
CI2(:,1:Nsamples-1) = [];
CQ2(:,1:Nsamples-1) = [];

S1 = CI1+j.*CQ1;
S2 = CI2+j.*CQ2;
```

Doppler_Detect.m

This function detects targets in the Doppler spectrum.

```
function [TL] = Doppler_Detect(D,vb,DT,v)

%Detection in Doppler
```

```
TL = zeros(length(D(1,:)),2);
TV = zeros(length(D(1,:)),2);

for i = 1:length(D(1,:))
    DoppThresh = DT*mean(D(:,i));

    MaxLoc = find(D(:,i) == max(D(:,i)));

    if MaxLoc ~= vb(1)-2 && MaxLoc ~= vb(length(vb))+2
        if D(MaxLoc,i) > DoppThresh
            TL(i) = MaxLoc;
        else
            TL(i) = 0;
        end
    end
end
```

Elevation_Angle.m

This function computes target elevation angle.

```
function [EAng] = Elevation_Angle(d,l,dPhi)

for b = 1:length(dPhi)
    for a = 0:5
        EA(a+1) = asind((a+dPhi(b)/360)*(l/d));
    end

    EA = EA(EA > 0);
    EA = EA(abs(EA) < 90);
```

```
    for i = 1:length(EA)
        EAng(b,i) = EA(i);
    end

    clear EA
end
```

Find_Discontinuity.m

This function locates discontinuities.

```
function [Discs] = Find_Discontinuity(W,Ns,Np,DT)

Discs = [];

dW = abs(W(1:Np-1,:) - W(2:Np,:));
dW = sum(dW');

Thresh = DT*mean(dW);

[dWpks,dWloc] = findpeaks(dW,'threshold',Thresh);

Discs = sort(dWloc);
```

Fix_Discontinuity.m

This function corrects discontinuities.

```
function [Wf] = Fix_Discontinuity(W,Ns,Np,DT,Discs)

%Fix Discontinouities in I/Q Data

Wf = zeros(Np,Ns);
```

```
if ~isempty(Discs)
    for k = 1:Ns
        Wf(1:Discs(1),k) = W(1:Discs(1),k);
        s(1) = W(Discs(1),k)-W(Discs(1)+2,k);
        i = 0;

        if length(Discs) > 1
            for i = 1:length(Discs)-1
                Wf(Discs(i)+1:Discs(i+1),k) =
                 W(Discs(i)+1:Discs(i+1),k)+sum(s(1:length(s)));
                Wf(Discs(i)+1) = Wf(Discs(i));

                s(i+1) = W(Discs(i+1),k)-W(Discs(i+1)+2,k);
            end
        end

        i=i+1;
        Wf(Discs(i)+1:length(Wf(:,1)),k) =
         W(Discs(i)+1:length(Wf(:,1)),k)+sum(s(1:length(s)));
        Wf(Discs(i)+1) = Wf(Discs(i));

        clear s
    end
else
    Wf = W;
end
```

LoadTDMS.m

This function loads data from *.tdms files into matlab.

```
function [I1a,Q1a,I2a,Q2a,Az,Time] = LoadTDMS(DF)

%Load TDMS File into Matlab

[TDMSdata]=convertTDMS(0,DF);

Time =
 TDMSdata.FileNameShort(1:length(TDMSdata.FileNameShort)-5);

for i = 1:4:length(TDMSdata.Data.MeasuredData(:))
    AzT = TDMSdata.Data.MeasuredData(i).Name;
    AzT(1) = [];
    AzT(length(AzT)-1:length(AzT)) = [];

    Az(1+(i-1)/4,1) = str2num(char(AzT));
    Az(1+(i-1)/4,2) = TDMSdata.Data.MeasuredData(i).Start_Time;

    clear AzT;
end

for i = 1:4:length(TDMSdata.Data.MeasuredData(:))
    I1a(:,1+(i-1)/4) = TDMSdata.Data.MeasuredData(i).Data;
    Q1a(:,1+(i-1)/4) = TDMSdata.Data.MeasuredData(i+1).Data;
    I2a(:,1+(i-1)/4) = TDMSdata.Data.MeasuredData(i+2).Data;
    Q2a(:,1+(i-1)/4) = TDMSdata.Data.MeasuredData(i+3).Data;
end
```

PhaseDifference.m

This function computes target channel phase difference.

```
function [dPhi] = PhaseDifference(S1,S2,P,PhiDiff)
```

```
%Compute Phase Difference

for i = 1:length(P)
    I1t = real(S1(:,P(i)));
    Q1t = imag(S1(:,P(i)));
    I2t = real(S2(:,P(i)));
    Q2t = imag(S2(:,P(i)));

    L = size(S1);
    L(2) = [];
    t = 1:L;

    I1fit = fit(t',I1t,'sin1');
    Q1fit = fit(t',Q1t,'sin1');
    I2fit = fit(t',I2t,'sin1');
    Q2fit = fit(t',Q2t,'sin1');

    dI = rad2deg(I1fit.c1-I2fit.c1);
    dQ = rad2deg(Q1fit.c1-Q2fit.c1);

    if dI < 0
        dI = dI+360;
    end

    if dQ < 0
        dQ = dQ+360;
    end

    dPhi(i,1) = (dI+dQ)/2;
    dPhi(i,2) = abs(dI-dQ);
```

```
end

dPhi(:,1) = dPhi(:,1)-PhiDiff;
```

polarmy.m

This function plots points in polar coordinates. It was not written by the author of this thesis. It can be found on the MATLAB Central File Exchange [18].

Remove_RXBlanking.m

This function removes portions of data during which the receiver is attenuated.

```
function [t,s] = Remove_RXBlanking(q)

b1 = 20;
b2 = 50;
IPP = 160;

pulse = [ones(1,b1) zeros(1,IPP-b1)];
blank = [];

for i = 1:floor(length(q)/IPP)
    blank = [blank pulse];
end

w = abs(xcorr(blank,q));

offset = length(q)-find(w == max(w))-1;

antiblank = [ones(1,offset)];

if offset > IPP-b2
```

```
    antiblank(1:offset-(IPP-b2)) = 0;
end

for i = 1:floor((length(q)-offset)/IPP)
    antiblank = [antiblank zeros(1,b2) ones(1,IPP-b2)];
end

antiblank(length(antiblank)+1:length(q)) = 0;

t = [];
s = [];
c = 1;

for i = 1:length(antiblank)
    if antiblank(i) == 1
        s = [s q(i)];
        t = [t i];
    end
end
```

RemoveFit.m

This function is the MTI.

```
function [Wf] = RemoveFit(W,Pd,Np,Ns)

%Fit and remove polynomial from data

for i = 1:Ns
    fr = polyfit([1:Np]',real(W(:,i)),Pd);
    fi = polyfit([1:Np]',imag(W(:,i)),Pd);
```

```
    Wfr(:,i) = real(W(:,i))-polyval(fr,[1:Np]');
    Wfi(:,i) = imag(W(:,i))-polyval(fi,[1:Np]');

    Wf(:,i) = Wfr(:,i)+j*Wfi(:,i);

    clear fr fi;
end
```

Target_Dopplers.m

This function computes a target's Doppler spectrum.

```
function [D,OD] = Target_Dopplers(S,P,Npulses,vb)

%Calculates Doppler Spectrum of Possible Targets

Dt = abs(fft(S(:,P)));

for i = 1:length(P)
    D(:,i) = fftshift(Dt(:,i));
end

for i = 1:size(D')
    D(vb,i) = sum(D(:,i))/Npulses;
end

for i = 1:length(D(:,1))/2
    OD(length(D(:,1))/2+1-i,:) =
     abs(abs(D(i,:)) - abs(D(length(D(:,1))+1-i,:)));
end
```

TriFilter.m

This function applies a detection filter to range data.

```
function [Wf,Wfn] = TriFilter(W,Pl,TFl,FiltWeight)

%Emphisize triangle result of pulse xcorr

for i = 1:length(W)
    FiltSum = W(i);

    for k = 1:TFl-1
        if i-k >= 1
            FiltSum =
             FiltSum-FiltWeight*abs(W(i-k)-((Pl-k)/Pl)^2*W(i));
        end

        if i+k <= length(W)
            FiltSum =
             FiltSum-FiltWeight*abs(W(i+k)-((Pl-k)/Pl)^2*W(i));
        end
    end

    Wfn(i) = FiltSum;

    Wf(i) = FiltSum;
end

Wf(1) = -0.1;           %First point occasionally is positive
```

A.2.3 Unused Functions

Calibrate_Detailed.m

This function extracts computes parameters from a data file.

```
function [I1a,Q1a,I2a,Q2a,dPhiCH] =
 Calibrate_Detailed(Cal_file,I1t,Q1t,I2t,Q2t,Az)

%Extract Cal Parameters

data = load(Cal_file)';

I1 = data(:,1);
Q1 = data(:,2);
I2 = data(:,3);
Q2 = data(:,4);

Ts = 10^-6;

I1 = I1';
Q1 = Q1';
I2 = I2';
Q2 = Q2';

I1s = sort(I1);
Q1s = sort(Q1);
I2s = sort(I2);
Q2s = sort(Q2);

I1max = mean(I1s(length(I1s)-4:length(I1s)));
Q1max = mean(Q1s(length(Q1s)-4:length(Q1s)));
I2max = mean(I2s(length(I2s)-4:length(I2s)));
```

```
Q2max = mean(Q2s(length(Q2s)-4:length(Q2s)));

I1min = mean(I1s(1:5));
Q1min = mean(Q1s(1:5));
I2min = mean(I2s(1:5));
Q2min = mean(Q2s(1:5));

I1_Amp = (I1max-I1min)/2;
Q1_Amp = (Q1max-Q1min)/2;
I2_Amp = (I2max-I2min)/2;
Q2_Amp = (Q2max-Q2min)/2;

%DC Offsets (V)
OI1 = (I1max+I1min)/2;
OQ1 = (Q1max+Q1min)/2;
OI2 = (I2max+I2min)/2;
OQ2 = (Q2max+Q2min)/2;

%IQ Amplitude Scale Factor (I = sc*Q)
sc1 = I1_Amp/Q1_Amp;
sc2 = I2_Amp/Q2_Amp;

%Channel Amplitude Scale (I1 = CHsc*I2)
CHsc = I1_Amp/I2_Amp;

t = [0:Ts:(length(I1)-1)*Ts];

I1fit = fit(t',I1','sin1');
Q1fit = fit(t',Q1','sin1');
I2fit = fit(t',I2','sin1');
Q2fit = fit(t',Q2','sin1');
```

```
I1f = I1fit.b1/(2*pi);
Q1f = Q1fit.b1/(2*pi);
I2f = I2fit.b1/(2*pi);
Q2f = Q2fit.b1/(2*pi);

%Baseband Doppler Frequencies
F1 = (I1f+Q1f)/2;
F2 = (I2f+Q2f)/2;

%IQ Phase Offsets Q = sin(theta + ph) (0 = 90 deg between I,Q)
Ph1 = I1fit.c1-Q1fit.c1-pi/2;
Ph2 = I2fit.c1-Q2fit.c1-pi/2;

%Channel Phase Offsets angle(I1) = angle(I2)+dPhiCH
dPhiCH = I1fit.c1-I2fit.c1;

%Calibrate Real Data
I1 = I1t;
Q1 = Q1t;
I2 = I2t;
Q2 = Q2t;

for i = 1:length(Az)
    I1a(:,i) = I1(:,i)-OI1;
    Q1a(:,i) =
     I1a(:,i)*tan(Ph1)+(Q1(:,i)-OQ1)/((1/sc1)*cos(Ph1));
    I2a(:,i) = CHsc*sc1/sc2*(I2(:,i)-OI2);
    Q2a(:,i) = I2a(:,i)*tan(Ph2)+
     CHsc*sc1/sc2*((Q2(:,i)-OQ2)/((1/sc2)*cos(Ph2)));
end
```

Clutter_Map.m

This function builds a clutter map from a folder of TDMS data files.

```
%Build Clutter Map

clear
clc

%Define Constants
Fc = 1.22e9;                    %TX Carrier Freq
Npulses = 128;                  %Number of Pulses per Az
Nsamples = 150;                 %Number of Samples per Pulse
ts = 1e-6;                      %Sampling Time (s)
tau = 8e-6;                     %Pulse Width (s)
PRF = 6250;                     %Pulse Repitition Freq (Hz)
Min_Dopp = 25;                  %Minimum Doppler Velocity (+-m/s)
d = 0.48;                       %Antenna Spacing (m)
PolyDeg = 2;                    %Polynomial Fit Degree
DiscThresh = 1.2;               %Discontinuity Threshold
TriFiltLength = 5;              %Length of Triangle Filter
TriFiltWeight = 0.9;            %Weight of Triangle Filter
DoppThresh = 10;                %Doppler Threshold

IPP = 1/PRF;                            %Inter Pulse Period (s)
Pulse = zeros(1,150);
Pl = round(tau/ts);                     %TX Pulse length in pts
Pulse(1:Pl) = 1;                        %TX Pulse
t = tau:ts:(Nsamples-1)*ts+tau;         %Time Vector (s)
Ranges = t.*((3e8)/2);                  %Range Vector (m)
f = linspace(-PRF/2,PRF/2,Npulses);     %Freq Vector (Hz)
v = -f.*(3e8)/(2*Fc);                   %Velocity Vector (m/s)
```

```
vb = find(v>-Min_Dopp & v<Min_Dopp);  %Velocity indexes to blank
vb(1) = [];                           %These ignored later
vb(1) = [];
vb(length(vb)) = [];
vb(length(vb)) = [];
l = (3e8)/Fc;                           %Wavelength (m)
CMap1 = zeros(360/Az_Tick,Nsamples);    %Clutter Map
CMap2 = zeros(360/Az_Tick,Nsamples);    %Clutter Map

%Select Data Folder
Data_Folder = uigetdir('C:\','Select Clutter Map Data Folder');
Data_File_List = dir([Data_Folder '\*.tdms']);

%Get Channel Phase Difference
PhiDiff = Calibrate();

FolderWait = waitbar(0,'Data Folder');
FileWait = waitbar(0,'Data File');

for df = 1:length(Data_File_List);
    waitbar(df/length(Data_File_List),q);

    [I1a,Q1a,I2a,Q2a,Az,Time] =
     LoadTDMS([Data_Folder '\' Data_File_List(df).name]);

    for a = 1:length(Az)
        waitbar(a/length(Az),h);

        for i = 1:Npulses
            I1(i,:) = I1a(1+Nsamples*(i-1):Nsamples*i,a);
            Q1(i,:) = Q1a(1+Nsamples*(i-1):Nsamples*i,a);
```

```
    I2(i,:) = I2a(1+Nsamples*(i-1):Nsamples*i,a);
    Q2(i,:) = Q2a(1+Nsamples*(i-1):Nsamples*i,a);
end

DI1 = Find_Discontinuity(I1,Nsamples,Npulses,DiscThresh);
DQ1 = Find_Discontinuity(Q1,Nsamples,Npulses,DiscThresh);
DI2 = Find_Discontinuity(I2,Nsamples,Npulses,DiscThresh);
DQ2 = Find_Discontinuity(Q2,Nsamples,Npulses,DiscThresh);

Discs = unique([DI1 DQ1 DI2 DQ2]);

I1 =
 Fix_Discontinuity(I1,Nsamples,Npulses,DiscThresh,Discs);
Q1 =
 Fix_Discontinuity(Q1,Nsamples,Npulses,DiscThresh,Discs);
I2 =
 Fix_Discontinuity(I2,Nsamples,Npulses,DiscThresh,Discs);
Q2 =
 Fix_Discontinuity(Q2,Nsamples,Npulses,DiscThresh,Discs);

[S1,S2] = Correlate(I1,Q1,I2,Q2,Pulse);

[S1f] = RemoveFit(S1,PolyDeg,Npulses,Nsamples);
[S2f] = RemoveFit(S2,PolyDeg,Npulses,Nsamples);

M = sum(abs(S1f.*S2f));
M1 = sum(abs(S1f));
M2 = sum(abs(S2f));

Mf = TriFilter(M,Pl,TriFiltLength,TriFiltWeight);
```

```
        P = find(Mf > 0);

        if length(P) >= 2
            for i = 1:length(P)-1
                if P(i)+1 == P(i+1)
                    if Mf(P(i)) > Mf(P(i+1))
                        P(i+1) = 0;
                    else
                        P(i) = 0;
                    end
                end
            end

            P(P == 0) = [];
        end

        CMap1(a,:) = CMap1(a,:) + M1;
        CMap2(a,:) = CMap2(a,:) + M2;
    end

end

close(h);
close(q);

CMap1 = CMap1/length(Data_File_List);
CMap2 = CMap2/length(Data_File_List);

save(['CMap_' Data_File_List(1).name '.mat'],'CMap1','CMap2')
```

A.3 Wiring Diagrams

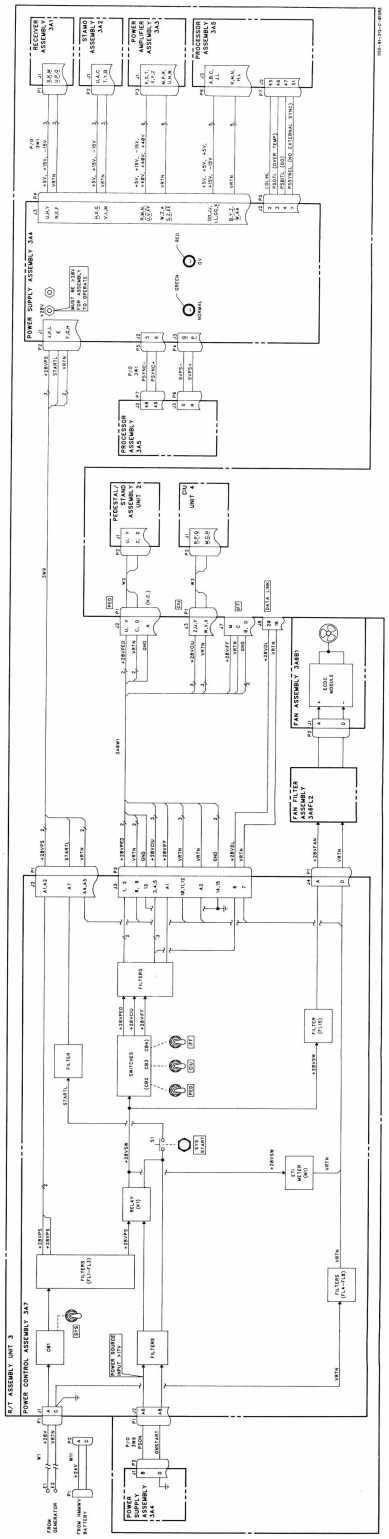

Figure A.20. PSTAR power distribution block diagram [15].

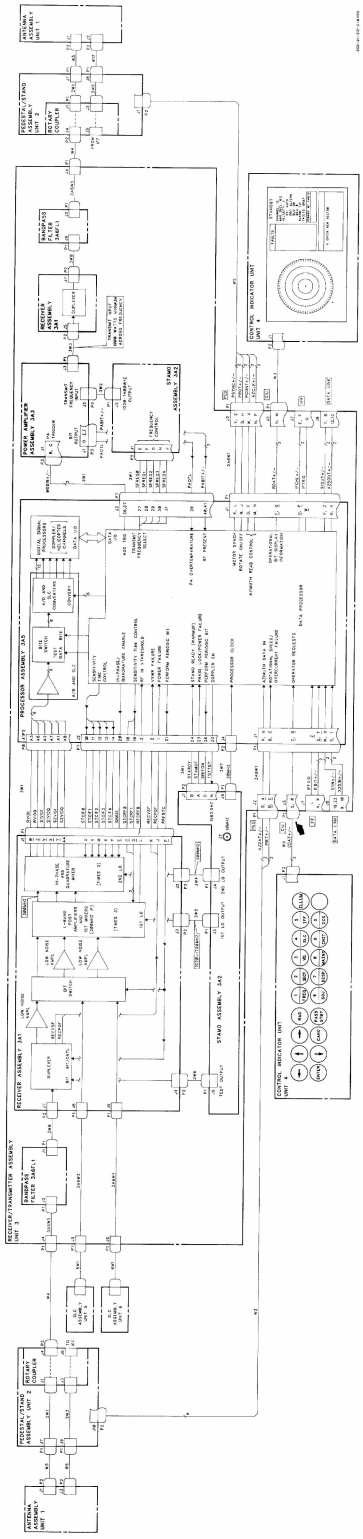

Figure A.21. PSTAR functional block diagram [15].